EXHIBIT D3

RTCA, Inc.
1150 18th Street, NW, Suite 910
Washington, D.C. 20036

Software Tool Qualification Considerations

RTCA DO-330
December 13, 2011

Prepared by: SC-205
© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Copies of this document may be obtained from

RTCA, Inc.

Telephone: 202-833-9339
Facsimile: 202-833-9434
Internet: www.rtca.org

Please visit the RTCA Online Store for document pricing and ordering information.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

FOREWORD

This report was prepared by RTCA Special Committee 205 (SC-205) and EUROCAE Working Group 71 (WG-71) and approved by the RTCA Program Management Committee (PMC) on December 13, 2011.

RTCA, Incorporated is a not-for-profit corporation formed to advance the art and science of aviation and aviation electronic systems for the benefit of the public. The organization functions as a Federal Advisory Committee and develops consensus-based recommendations on contemporary aviation issues. RTCA's objectives include but are not limited to:

- coalescing aviation system user and provider technical requirements in a manner that helps government and industry meet their mutual objectives and responsibilities;
- analyzing and recommending solutions to the system technical issues that aviation faces as it continues to pursue increased safety, system capacity, and efficiency;
- developing consensus on the application of pertinent technology to fulfill user and provider requirements, including development of minimum operational performance standards for electronic systems and equipment that support aviation; and
- assisting in developing the appropriate technical material upon which positions for the International Civil Aviation Organization and the International Telecommunication Union and other appropriate international organizations can be based.

The organization's recommendations are often used as the basis for government and private sector decisions as well as the foundation for many Federal Aviation Administration Technical Standard Orders.

Since the RTCA is not an official agency of the United States Government, its recommendations may not be regarded as statements of official government policy unless so enunciated by the U.S. government organization or agency having statutory jurisdiction over any matters to which the recommendations relate.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

This Page Intentionally Left Blank

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

TABLE OF CONTENTS

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.
© 2011 RTCA, Inc.

LIST OF FIGURES

LIST OF TABLES

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

1.0 INTRODUCTION

1.1 Purpose

The purpose of this document is to provide tool qualification guidance. Additionally, clarification material is provided in the form of Frequently Asked Questions (FAQs).

1.2 Scope

Software tools are widely used in multiple domains, to assist in developing, verifying, and controlling other software. In the context of this document a tool is a computer program or a functional part thereof, used to help develop, transform, test, analyze, produce, or modify another program, its data, or its documentation. Examples are automated code generators, compilers, test tools, and modification management tools. This document explains the process and objectives for qualifying tools.

This document was developed for the following reasons:

a. Tools are different from the software using the tools and form a unique domain; therefore, tool-specific guidance for both tool developers and tool users is needed.

b. Tools are often developed by teams other than those who use the tools to develop software. These tool development teams frequently do not have software guidance background (examples of guidance include DO-178C or DO-278A). This tool-specific document benefits tool development teams and helps them avoid confusion and misinterpretation.

c. This document provides guidance for airborne and ground-based software. It may also be used by other domains, such as automotive, space, systems, electronic hardware, aeronautical databases, and safety assessment processes.

1.3 How to Use This Document

When a domain or a project desires to use this document, the following should be considered.

a. To offer maximum applicability and flexibility, five tool qualification levels (TQLs) are defined; however, not all levels may be applicable for every domain. Each domain should determine the applicability of the TQLs for their particular needs. Please reference the appropriate domain-specific document (such as an RTCA, SAE, or EUROCAE document) to determine if, and to what level, a specific tool needs to be qualified. As an example, Appendix B shows how the DO-278A domain defines the tool qualification criteria and the applicable TQL for ground-based communication, navigation, surveillance, and air traffic management (CNS/ATM) software. However, each domain may use different criteria. For example, they may not use all TQLs; they may have different tool criteria which map to the used TQLs; or they may adapt some of the objectives of this document for their particular applicability.

b. Throughout this document terms such as “software life cycle”, “software processes”, “software plans”, and “software” are used to refer to the product life cycle, processes, plans, and domain where the tool will be used (that is, a software domain is used instead of a generic domain). For other domains the word “software” may be replaced

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.
© 2011 RTCA, Inc.

by the appropriate domain, such as, "electronic hardware", "system", "aeronautical database", "aviation software", etc.

c. The term "certification authority" is used as a generic term to identify the entity in charge of the approval of the tool qualification data. Likewise, the term "certification" is used generically to refer to the approval of the product using the tool by the applicable authorities. The term "applicant" is also used to identify the entity seeking software approval and/or tool qualification.

d. Possible intentions, interpretations, and adaptations of this document should be included in the appropriate plans in order to obtain approval by the appropriate certification authority.

e. Coordination between the tool life cycle and the software life cycle should be executed and documented in the following software life cycle data:

- A plan identifying the tools and their need for qualification. This data is named "Plan for Software Aspects of Certification" (PSAC) throughout this document.
- A summary and status of all activities performed, including the status of the tools to be qualified. This data is named "Software Accomplishment Summary" (SAS) throughout this document.
- A configuration index of the software environment, including the identification of the tools and the tool qualification data. This data is named "Software Life Cycle Environment Configuration Index" (SECI) throughout this document.

Regardless of the domain, equivalent data (which may have different titles) are expected to be produced.

f. In this document, it is assumed that the tool is used to eliminate, reduce, or automate a software life cycle process without the tool's output being verified, and that these processes are defined in the form of a set of objectives. Therefore, the certification credit of the tool is associated with one or several objectives. It is possible that the guidance for a domain is not defined in the form of "processes" and "objectives". In this case, the definition of certification credit should be adapted to the domain terminology.

g. For a domain where no guidance exists to define the tool qualification criteria and applicable TQL, a project in this domain may propose to use this document. The applicable TQL and its rationale should be defined in the project's PSAC or equivalent document.

h. Users of this document should reference Annex A in order to determine applicability of objectives for each TQL.

1.4 Document Overview

Figure 1-1 provides an overview of this document. It should be noted that sections 1, 2, and 3, as well as the appendices contain explanatory information to aid the reader in understanding this document's purpose (these sections are not considered to be guidance).

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

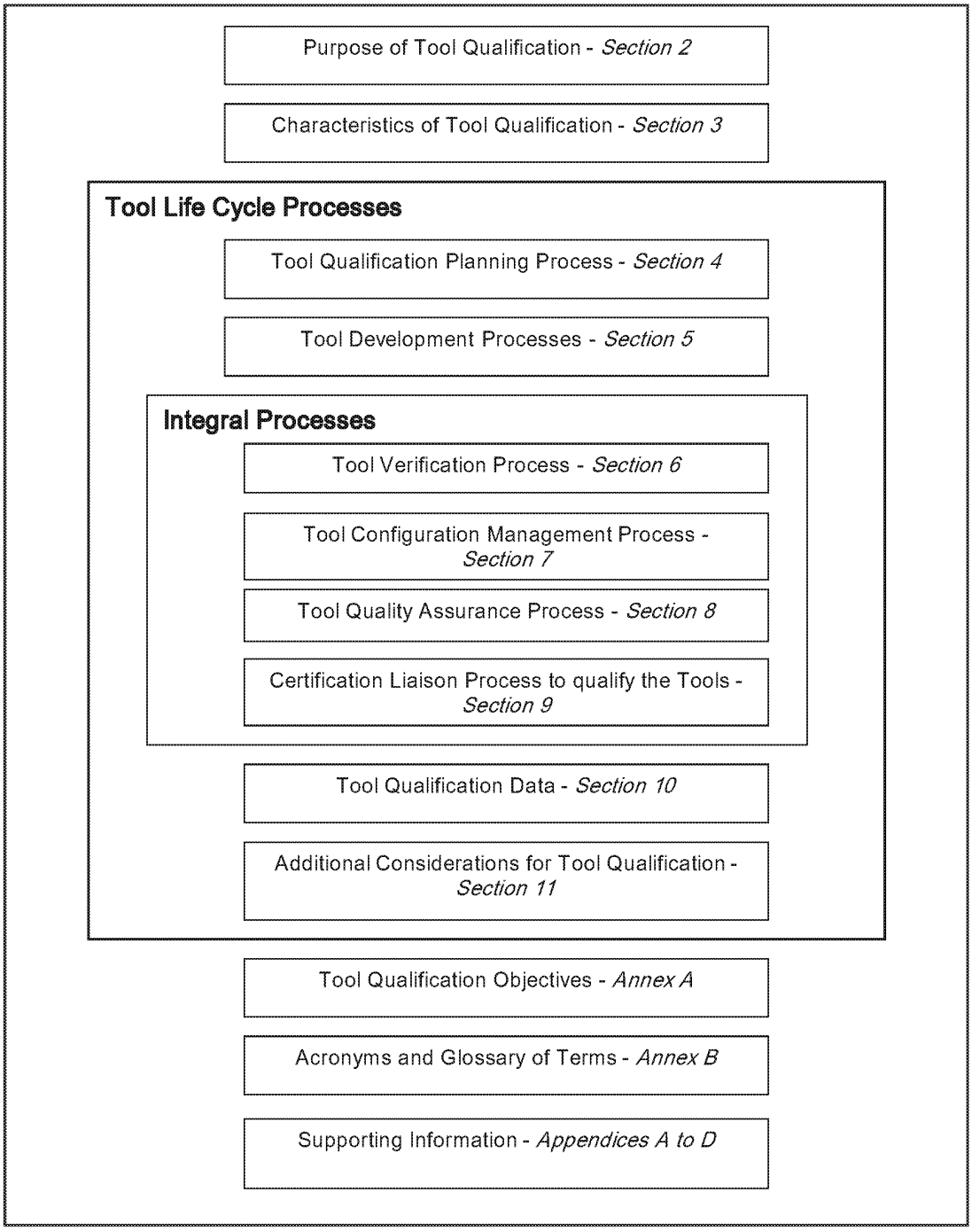


Figure 1-1 Document Overview

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.
© 2011 RTCA, Inc.

This Page Intentionally Left Blank

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

2.0 PURPOSE OF TOOL QUALIFICATION

Tools can assist in software development to analyze and potentially improve system safety by the automation of the activities performed and by predictably performing functions that may be prone to human error. However, an error in the tool may have a negative impact on software functionality if the tool inadequately performs its intended functions. In order to avoid this risk and to ensure the integrity of the tool functionality, the tool should be developed and verified using adequate processes. Tool qualification serves this purpose.

Tool qualification is the process necessary to obtain certification credit for a tool. This credit may only be granted within the context of a project requiring approval.

The purpose of the tool qualification process is to obtain confidence in the tool functionality. The tool qualification effort varies based upon the potential impact that a tool error could have on the system safety and upon the overall use of the tool in the software life cycle process. The higher the risk of a tool error adversely affecting system safety, the higher the rigor required for tool qualification.

The tool qualification process may be applied to one or more functions in a tool, a single tool, or a collection of tools (see 11.1 for additional information on multi-function tools).

During the qualification effort, the output of all qualified tool functions should be shown to be correct using the objectives of this document. For a tool whose output may vary within expectations, it should be shown that the variation does not adversely affect the intended use of the output and that the correctness of the output can be established. However, for a tool whose output is part of the software and thus could insert an error, it should be demonstrated that qualified tool functions produce the same output for the same input data when operating in the same environment.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

This Page Intentionally Left Blank

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

3.0 CHARACTERISTICS OF TOOL QUALIFICATION

3.1 Qualification Levels

The required qualification level of a tool is based on the tool use and its potential impact in the software life cycle process. This document identifies five Tool Qualification Levels (TQL-1 to TQL-5). As the potential impact of the tool on the software life cycle process increases, the TQL rigor increases due to the potential impact on system safety. The determination of which level is assigned to the tool qualification is determined by an assessment of the tool use in the software life cycle process as defined in the domain related guidance (for example, section 12.2 of DO-178C).

The level of qualification rigor varies according to qualification level. TQL-1 is the most rigorous level and requires well defined and executed tool development, verification, and integral processes with the highest degree of verification independence. The remaining levels require decreasing amounts of rigor. TQL-5 is the least rigorous qualification level. Annex A defines the applicable objectives for each TQL.

3.2 Qualification Stakeholders

Tool qualification typically involves multiple stakeholders. In most projects there will be both a tool user and a tool developer. The tool user typically identifies the tool to be used, assesses its impact on the software processes, addresses the use of the tool in the scope of the software process in which the tool is used, and performs the tool qualification within the context of the software approval. The tool developer typically describes the processes of the tool development, verification, and integral processes, and addresses the development of the tool in compliance with the user needs expressed in the Tool Operational Requirements (TOR).

The typical stakeholders and roles identified above may vary. Likewise, additional stakeholders may be identified (for example, a tool integrator or a tool verifier). The planning documents, PSAC, and/or Tool Qualification Plan (TQP) (as appropriate) should identify the stakeholders and the roles and responsibilities of each stakeholder for the tool qualification effort. The applicant is ultimately responsible for ensuring that all of the tool qualification objectives are satisfied.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

This Page Intentionally Left Blank

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

4.0 TOOL QUALIFICATION PLANNING PROCESS

This section discusses the objectives and activities of the tool qualification planning process. During the tool qualification planning process, the tool is identified; its impact in the software life cycle process is assessed; and the tool development, verification, and integral processes of the complete tool life cycle are described.

The tool qualification planning process includes: (1) the tool qualification assessment as part of the software planning process (see "Planning Process" portion of Annex A Table T-0), and (2) the tool planning process (see all of Annex A all of Table T-1). Annex A Table T-0 and Table T-1 summarize the objectives and outputs of the tool qualification planning process for each TQL.

4.1 Determining the Tool Qualification Needs

During the software planning process:

- the tools used in the scope of the software life cycle process are identified;
- each tool's intended use is described;
- the need for tool qualification is defined;
- the TQLs are determined;
- the tool qualification stakeholders and their assigned roles and responsibilities are identified; and
- the tool operational environment is described.

If there are multiple tool operational environments, then all such environments are described. All these items should be discussed in the additional considerations section of the PSAC. The PSAC explains the use of tools in the software life cycle processes and the need for tool qualification. See section 10.1.1 of this document for the tool aspects that should be addressed in the PSAC.

4.2 Tool Life Cycle Definition

During the tool planning process, a project defines one or more life cycle(s) by choosing the activities for each process, specifying a sequence for the activities, and assigning responsibilities for the activities.

The tool life cycle processes are:

a. The tool planning process that defines and coordinates the activities of the tool development and integral processes.

b. The tool development processes that produce the tool.

c. The integral processes that ensure the correctness, control, and confidence of the tool life cycle processes and their outputs. The integral processes are the tool verification process, the tool configuration management process, the tool quality assurance process, and the qualification liaison process. It is important to understand that the

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

integral processes are performed concurrently with the tool development processes throughout the tool life cycle.

This document does not prescribe preferred tool life cycles and interactions between them. The separation of the processes is not intended to imply a structure for the organization(s) that performs them. For each tool, the tool life cycle(s) includes these processes.

The tool life cycle definition includes the transition criteria, which are used to determine whether a process may be entered or re-entered. Each process performs activities on inputs to produce outputs. A process may produce feedback to other processes and receive feedback from others. The definition of feedback includes how information is recognized, controlled, and resolved by the receiving process. An example of feedback is problem reporting.

The transition criteria will depend on the planned sequence of tool development processes and integral processes, and may be affected by the TQL. If the transition criteria established for the process are satisfied, every input to a process need not be complete before that process can be initiated.

If a process acts on partial inputs, the inputs to the process should be examined to ensure that they meet the transition criteria. Also, subsequent inputs to the process should be examined to determine that the previous outputs of the tool development and tool verification processes are still valid.

4.3 Tool Planning Process Objectives

The objectives of the tool planning process are:

a. Define the tool life cycle process(es) in compliance with its intended use as defined in the PSAC, including the items specified in section 4.1 above.

b. Determine the tool's life cycle, including the inter-relationships between the processes, their sequencing, feedback mechanisms, and transition criteria.

c. Identify the tool development environment, including the methods and tools to be used for the activities of each tool life cycle process.

d. If necessary, address additional considerations, such as those discussed in section 11 of this document and the need to qualify any tool(s) used in the framework of the tool life cycle processes.

e. Define tool development standards.

f. Ensure that tool plans comply with section 10 of this document.

g. Coordinate the development and revision of the tool plans.

4.4 Tool Planning Process Activities

In order to satisfy the objectives of this document, effective planning is needed. Activities for this process include:

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

a. A TQP, Tool Development Plan, Tool Verification Plan, Tool Quality Assurance Plan, and Tool Configuration Management Plan that cover the complete tool life cycle should be developed. These plans may be separate or combined, as appropriate for the specific project needs. As an example, a tool supplier may propose a qualifiable tool and develop plans that cover only the tool development process (see section 11.3 for commercial-off-the-shelf (COTS) tool qualification), but the description of the Tool Operational Requirements definition and tool operational integration processes may be addressed in a separate set of tool plans. Each plan and the supporting standards are briefly described below and detailed in section 10.

    1. The TQP (see 10.1.2) serves as the primary means for communicating the proposed tool life cycle and development methods to the certification authority for agreement. The TQP defines the means of compliance with this document and any other applicable guidance (for example, a domain or project-specific document).

    2. The Tool Development Plan (see 10.1.3) defines the tool life cycle, tool development environment and processes, and the means by which the tool development process objectives will be satisfied.

    3. The Tool Verification Plan (see 10.1.4) defines the means by which the tool verification process objectives will be satisfied.

    4. The Tool Configuration Management Plan (see 10.1.5) defines the means by which the tool configuration management process objectives will be satisfied.

    5. The Tool Quality Assurance Plan (see 10.1.6) defines the means by which the tool quality assurance process objectives will be satisfied.

b. Tool development standards should be developed. The purpose of these standards is to define the rules and constraints for the tool development processes. The tool development standards include the Tool Requirements Standards (see 10.1.7), the Tool Design Standards (see 10.1.8), and the Tool Code Standards (see 10.1.9). The tool verification process uses these standards as a basis for evaluating the compliance of actual outputs of a process with intended outputs.

c. The tool verification environment(s) should be defined. The verification environment(s) should be shown to be representative of the tool operational environment. Care should be taken to identify the operating system, additional hardware, additional software components, etc. It should be noted that it may be necessary to have several tool verification environments if there are potentially multiple tool operational environments (for example, a tool may be used on multiple platforms with different versions of operating systems, depending on the specific operational needs).

d. Reviews of the tool planning process should be conducted to ensure that the plans and standards comply with the guidance of this document and the means to execute them exist.

e. An assessment on all the tools used in the framework of the tool life cycle processes should be conducted in order to identify the need for qualification of these tools. Qualification of these tools is needed when processes of this document are eliminated, reduced, or automated by the use of a tool without its output verified as

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

specified in section 6. For a tool that can introduce an error in the outputs of a tool, the applicable TQL is the same as the tool being developed. For a tool that cannot introduce an error in the output of the tool, but may fail to detect an error in the tool life cycle data, the applicable TQL is TQL-5.

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

5.0 TOOL DEVELOPMENT LIFE CYCLE AND PROCESSES

This section discusses the objectives and activities of the tool development life cycle. The tool development processes are applied as defined by the planning process (see 4) and the Tool Development Plan (see 10.1.3). The tool development life cycle for a specific tool may be tailored based on the TQL. The primary purpose of the tool qualification process is to confirm that a tool meets its requirements and is compliant with the software life cycle process needs. Therefore, the needs from the software life cycle process perspective are defined in the Tool Operational Requirements and the description of all the functionality and characteristics necessary to develop the tool is defined in the Tool Requirements. The overall tool development life cycle includes three major processes, as illustrated in Figure 5-1 and described below. Although these processes are described in the context of a linear waterfall life cycle model, it should be noted that other life cycle models may be used.

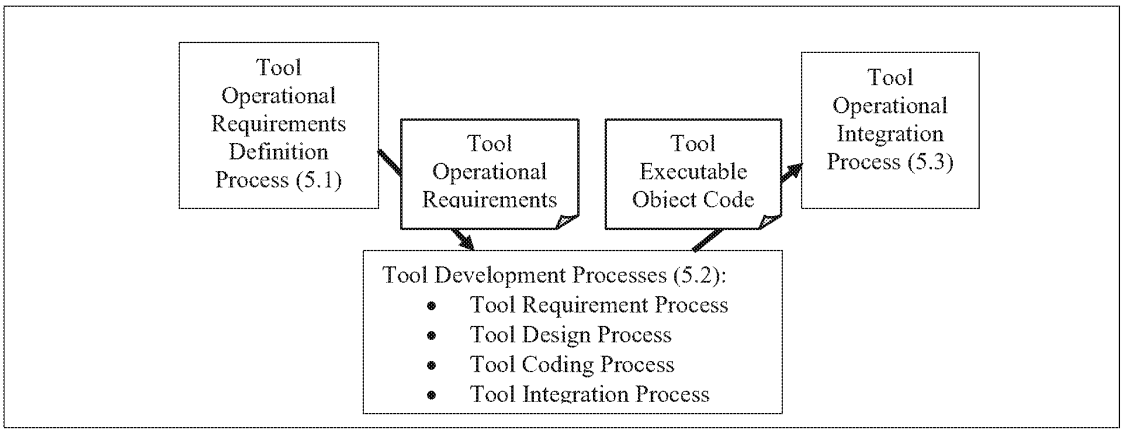


FIGURE 5-1
OVERALL TOOL DEVELOPMENT LIFE CYCLE

Each process is briefly described here and detailed later in sections 5.1 to 5.3.

a. Tool Operational Requirements Definition Process (see 5.1): The Tool Operational Requirements should identify how the tool is to be used within the software life cycle process. It may not provide all requirements necessary to develop the tool. The amount of detail of the Tool Operational Requirements is dependent on the tool and its intended use. Validation and verification of the Tool Operational Requirements are necessary to confirm that the tool satisfies the needs of the software life cycle process. The Tool Operational Requirements definition process is described in the Tool Development Plan. The output of this process is the Tool Operational Requirements.

b. Tool Development Processes (see 5.2): The tool development processes are applied as defined by the TQP and the Tool Development Plan. The tool development processes include:

   - Tool requirements process.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.
© 2011 RTCA, Inc.

- Tool design process.
- Tool coding process.
- Tool integration process.

The tool requirements process produces the Tool Requirements. Tool Requirements are the requirements used to develop the tool. The Tool Requirements should be produced from analysis of the Tool Operational Requirements. However, some tools will be developed for a wide variety of applications and may not be based on any specific Tool Operational Requirements. See section 11.3 for more information about these tools.

The Tool Design Description includes the tool architecture and may generate one or more levels of low-level tool requirements. However, if the Tool Source Code can be generated directly from Tool Requirements then the Tool Requirements are also considered as low-level requirements, and the guidance for low-level requirements also apply to the Tool Requirements. The Source Code is implemented from the tool architecture and the low-level tool requirements.

The Tool Executable Object Code is generated in the tool development environment.

Table T-2 in Annex A is a summary of the objectives and outputs of the tool development processes by TQL.

c. Tool Operational Integration Process (see 5.3): The tool operational integration process is applied as defined by the tool plans. Inputs to the tool operational integration process are the Tool Executable Object Code, the user instructions, and the installation instructions. They are provided by the tool development process for the tool use and installation into the tool operational environment. The output of the tool operational integration process is the Tool Executable Object Code installed in the tool operational environment.

5.1 Tool Operational Requirements Definition Process

The Tool Operational Requirements identify how the tool is to be used within the software life cycle process. The software plans and the tool plans are the inputs to this process. The Tool Operational Requirements are the primary outputs of this process (see 10.3.1). The "Tool Operational Requirements Process" portion of Table T-0 in Annex A summarizes the objective and output of this process by TQL.

5.1.1 Tool Operational Requirements Objectives

The objective of the tool operational requirements process is:

a. Tool Operational Requirements are defined.

5.1.2 Tool Operational Requirements Activities

Activities for this process include:

a. Tool Operational Requirements should be developed in order to include, as a minimum, all items defined in section 10.3.1.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

b. Tool Operational Requirements should be verifiable and consistent. They should include enough detail to demonstrate that the functionality and the outputs of the tool correspond to the identified software life cycle activities to be performed by the tool.

c. Tool Operational Requirements should provide enough detail to support the verification of the tool's capability to justify taking credit for satisfying the process(es) automated, eliminated, or reduced.

5.2 Tool Development Processes

The purpose of the tool development processes is to define the Tool Requirements and construct the software necessary for their implementation. The typical tool development processes are:

- Tool requirements process.
- Tool design process.
- Tool coding process.
- Tool integration process.

Table T-2 of Annex A summarizes the objectives and outputs of the tool development processes by TQL.

5.2.1 Tool Requirements Process

The Tool Requirements are the requirements used to develop and verify the tool. Tool Operational Requirements, Tool Development Plan, and Tool Requirements Standards are the inputs to this process. The Tool Requirements are the primary outputs of this process (see 10.2.1).

5.2.1.1 Tool Requirements Objectives

The objectives of the tool requirements process are:

a. Tool Requirements are developed.

b. Derived tool requirements are defined, if needed.

5.2.1.2 Tool Requirements Activities

Activities for this process include:

a. The Tool Requirements should be developed and documented in a manner that provides a determination if the tool satisfies the Tool Operational Requirements.

b. The Tool Requirements should include all functional and interface-related requirements that will be implemented in the tool.

c. The Tool Requirements should be developed following the processes described in the Tool Development Plan.

d. The Tool Requirements should be developed using the Tool Requirements Standards.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

e. The Tool Requirements should be verifiable and consistent.

f. The Tool Requirements should be defined such that abnormal behavior is detected and that invalid output is prevented. For example, it is better for the tool to produce no output than to produce wrong results. Therefore, the Tool Requirements should address failure modes and describe responses to failure conditions.

g. Each Tool Requirement should trace to one or more Tool Operational Requirements, with the exception of derived tool requirements.

h. Derived tool requirements are those not traceable to Tool Operational Requirements. The existence of derived tool requirements should be justified, and they should be evaluated to ensure that they do not negatively impact the expected functionality and outputs defined in the Tool Operational Requirements.

i. The Tool Requirements should provide user instructions, list of error messages, and constraints.

j. The Tool Requirements should identify requirements related to unused tool functions that do not trace to Tool Operational Requirements. See section 11.1 for guidance on multi-function tools.

k. The Tool Requirements should be defined to a level of detail appropriate to ensure proper implementation and to assess correctness of the tool (for example, defining underlying models or mathematical theories).

5.2.2 Tool Design Process

Tool Requirements are refined through one or more iterations in the tool design process to develop the tool architecture and low-level tool requirements until it can be used to implement Tool Source Code. The tool design process inputs are the Tool Requirements, the Tool Development Plan, and the Tool Design Standards. When the planned transition criteria have been satisfied, the Tool Requirements are used in the tool design process to develop the tool architecture and low-level tool requirements. The primary output of the tool design process is the Tool Design Description (see 10.2.2).

5.2.2.1 Tool Design Objectives

The Tool Design Description is produced during the tool design process. The objectives are:

a. The tool architecture is developed.

b. Low-level tool requirements are developed.

c. Derived low-level tool requirements are defined, if needed.

5.2.2.2 Tool Design Activities

Activities for this process include:

a. The tool design process should define the tool architecture.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

b. The tool design should address all tool architecture features, such as protection, if applicable. Protection may be used in order to isolate parts of the tool or the collection of tools in order to apply a different approach or different level of qualification. See section 11.1 for multi-function tools.

c. The Tool Requirements should be refined into low-level tool requirements that are traceable to Tool Requirements or identified as derived low-level tool requirements.

d. Derived low-level tool requirements are not traceable to Tool Requirements. The existence of derived requirements should be justified, and they should be evaluated to ensure that they do not impact the expected functionality and outputs defined in the Tool Operational Requirements.

e. The Tool Design Description, including tool architecture and low-level tool requirements, should conform to the Tool Design Standards.

f. The tool design should be verifiable and consistent.

g. Where external components are utilized by the tool, which are not under control of the developer of the tool (such as operating system functions or an external software library), the interfaces to these external components utilized by the tool should be identified in the Tool Design Description. The description of the interface should identify all the external components, such as file management routines, primitives, memory allocation calls, and routines supporting the user interface management (for example, command line or display message).

5.2.3 Tool Coding Process

In the tool coding process, the Tool Source Code is implemented from the Tool Design Description, including tool architecture and low-level tool requirements. The tool coding process inputs are the Tool Design Description, the Tool Development Plan, and the Tool Code Standards. The tool coding process may be entered or re-entered when the planned transition criteria are satisfied. The primary output of this process is the Tool Source Code (see 10.2.3).

5.2.3.1 Tool Coding Objectives

The objective of the tool coding process is:

a. Tool Source Code is developed from the low-level tool requirements.

5.2.3.2 Tool Coding Activities

Activities for this process include:

a. The Tool Source Code should implement the low-level tool requirements and conform to the tool architecture.

b. The Tool Source Code should conform to the Tool Code Standards.

c. The Tool Source Code should be traceable to the low-level tool requirements.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

d. Inadequate or incorrect inputs detected during the tool coding process should be provided to the process(es) that produced the incorrect input(s) for clarification or correction.

5.2.4 Tool Integration Process

The tool integration process consists of producing the tool in its executable format. This is classically achieved by compiling and linking the Tool Source Code in the tool development environment. If the tool development environment is different than the tool verification environment, then the Tool Executable Object Code should then be installed in the tool verification environment. The tool integration process may be entered or re-entered when the planned transition criteria have been satisfied. The tool integration process input is the Tool Source Code and the Tool Development Plan. The primary outputs of the tool integration process are the Tool Executable Object Code (see 10.2.4) and the compiler and linking data.

5.2.4.1 Tool Integration Objectives

The objectives of the tool integration process are:

a. Tool Executable Object Code is generated in the tool development environment.

b. Tool Executable Object Code is installed into the tool verification environment(s).

5.2.4.2 Tool Integration Activities

Activities for this process include:

a. The Tool Executable Object Code should be generated from the Tool Source Code and the compiler and linking instructions. It should be noted that the Tool Executable Object Code sometimes includes other files, such as configuration files.

b. Inadequate or incorrect inputs detected during the tool integration process should be provided to the process(es) that produced the inadequate or incorrect input(s).

c. If the tool verification environment is different than the tool development environment, or if there are multiple tool verification environments, then the Tool Executable Object Code should be installed in the tool verification environment(s).

5.2.5 Tool Development Process Traceability

Tool development process traceability activities include:

a. Trace Data, showing the bi-directional association between Tool Operational Requirements and Tool Requirements, is developed to:

   1. Enable verification of the complete implementation of the Tool Operational Requirements.

   2. Give visibility to those derived requirements that are not directly traceable to Tool Operational Requirements.

b. Trace Data, showing the bi-directional association between the Tool Requirements and low-level tool requirements, is developed to:

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

1. Enable verification of the complete implementation of the Tool Requirements.

2. Give visibility to those derived low-level tool requirements that are not directly traceable to Tool Requirements and to the architectural design decisions made during the tool design process.

c. Trace Data, showing the bi-directional association between low-level tool requirements and Tool Source Code, is developed to:

1. Enable verification that no Tool Source Code implements an undocumented function.

2. Enable verification of the complete implementation of the low-level tool requirements.

5.3 Tool Operational Integration Process

The purpose of the tool operational integration process is to install the Tool Executable Object Code in the tool operational environment identified in the Tool Operational Requirements and any additional instructions (such as a user's manual), and to gain confidence that the tool is operating as expected by the tool user in the tool operational environment.

The tool operational integration process may be entered or re-entered when the planned transition criteria have been satisfied. The tool operational integration process inputs are the Tool Executable Object Code (including all associated files, if necessary) and the user manual (which may be part of the Tool Requirements).

The primary outputs of the tool operational integration process are the Tool Executable Object Code (see 10.2.4) installed into the tool operational environment and a Tool Installation Report (see 10.3.2). The "Tool Operational Integration Process" portion of Table T-0 in Annex A contains a summary of the objective and output of this process by TQL.

5.3.1 Tool Operational Integration Process Objectives

The objective of the tool operational integration process is:

a. Tool Executable Object Code is installed into the tool operational environment.

5.3.2 Tool Operational Integration Activities

Activities for this process include:

a. The Tool Executable Object Code (including all associated files, such as, configuration files) should be installed into the tool operational environment.

b. Inadequate or incorrect inputs detected during the tool operational integration process should be provided to the process(es) that produced the inadequate or incorrect input(s).

c. A Tool Installation Report should be generated (see 10.3.2).

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

This Page Intentionally Left Blank

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

6.0 TOOL VERIFICATION PROCESS

This section discusses the objectives and describes activities for the overall tool verification process. This process involves two parts:

a. The tool verification process (see 6.1).

b. The tool operational verification and validation process (see 6.2).

Note: The verification of the outputs of the planning process is addressed in section 4.4.d.

6.1 Tool Verification Process

Verification is the technical assessment of the outputs of both the tool development processes and the tool verification process. The tool verification process is applied as defined by the tool planning process (section 4) and the Tool Verification Plan (see 10.1.4). Verification includes reviews, analyses, and tests.

Tables T-3 through T-7 of Annex A contain a summary of the objectives and outputs of the tool verification process by TQL.

6.1.1 Purpose of Tool Verification Process

The purpose of the tool verification process is to detect and report errors that may have been introduced during the tool development processes. Removal of the errors is an activity of the tool development processes. The tool verification process verifies that:

a. The Tool Operational Requirements have been implemented into the Tool Requirements and that the Tool Requirements satisfy and trace to the Tool Operational Requirements.

b. The Tool Requirements have been implemented into the Tool Design Description (tool architecture and low-level tool requirements) which satisfies the Tool Requirements. Low-level tool requirements should satisfy and trace to the Tool Requirements.

c. The tool architecture and low-level tool requirements have been developed into the Tool Source Code which satisfies the Tool Design Description. Tool Source Code should satisfy and trace to the tool low-level requirements.

d. The Tool Executable Object Code satisfies the tool's requirements (that is, intended function) and provides confidence in the absence of unintended functionality.

6.1.2 Overview of Tool Verification Activities

Tool verification objectives are satisfied through a combination of reviews, analyses, the development of test cases and procedures, and the subsequent execution of those test procedures. Reviews and analyses provide an assessment of the accuracy, completeness, and verifiability of the Tool Requirements, Tool Design Description, and Tool Source Code. Reviews also assess compliance to the applicable development standards. The development of test cases and procedures may provide further assessment of the internal consistency and completeness of the Tool Requirements and low-level tool requirements.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

The execution of the test procedures provides a demonstration of compliance of the Tool Executable Object Code with the Tool Requirements and the low-level tool requirements.

The inputs to the tool verification process include the Tool Operational Requirements, Tool Requirements, Tool Design Description, Trace Data, Tool Source Code, Tool Executable Object Code, and Tool Verification Plan.

The outputs of the tool verification process are recorded in Tool Verification Cases and Procedures (see 10.2.5), Tool Verification Results (see 10.2.6), and the associated Trace Data (see 10.2.7).

Considerations for this process include:

a. In order to determine that the higher level of requirements is satisfied by the lower levels of requirements (or source code) and to identify derived requirements, traceability should exist as follows:

    1. Tool Requirements should be traceable to the Tool Operational Requirements.

    2. The low-level tool requirements should be traceable to the Tool Requirements.

    3. The Tool Source Code should be traceable to the low-level tool requirements.

b. In order to identify the requirements not covered by tests cases, the test cases should be traceable to the requirements (Tool Requirements and low-level tool requirements).

c. In order to verify all the tests cases are implemented in test procedures, test procedures should be traceable to tests cases.

d. When it is not possible to verify specific Tool Requirements, other means should be provided and their justification for satisfying the tool verification process objectives defined in the Tool Verification Plan and/or Tool Verification Results.

e. Deficiencies and errors discovered during the tool verification process should be reported to the appropriate tool development processes for clarification and correction.

6.1.3 Reviews and Analyses

Reviews and analyses should be performed to detect and report errors that may have been introduced during the tool development processes.

6.1.3.1 Reviews and Analyses of the Tool Requirements

The purpose of these reviews and analyses is to detect and report requirements errors that may have been introduced during the tool requirements process. These reviews and analyses activities confirm that the Tool Requirements satisfy these objectives:

a. Compliance with Tool Operational Requirements: The objective is to ensure that all the Tool Operational Requirements are implemented into the Tool Requirements and that derived tool requirements, and the reason for their existence, are correctly defined.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

b. Accuracy and consistency: The objective is to ensure that each Tool Requirement is accurate, unambiguous, and sufficiently detailed and that the requirements do not conflict with each other. This objective includes verification that the level of detail of the Tool Requirements is appropriate to ensure proper implementation and to assess correctness of the tool (see 5.2.1.2.k).

c. Compatibility with the tool operational environment: The objective is to ensure that if constraints exist to address the compatibility with the tool operational environment, then compatibility requirements are defined.

d. Failure modes and errors: The objective is to ensure that the Tool Requirements define the behavior of the tool in response to error conditions and specific requirements addressing failure modes and incorrect inputs are identified.

e. User information: The objective is to ensure that the Tool Requirements provide user instructions, list of error messages, and constraints.

f. Verifiability: The objective is to ensure that the Tool Requirements can be verified.

g. Conformance to standards: The objective is to ensure that the Tool Requirements Standards were followed during the tool requirements process and that deviations from the standards are justified.

h. Traceability: The objective is to ensure that the Tool Operational Requirements were developed into the Tool Requirements.

i. Algorithm aspects: The objective is to ensure the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities.

6.1.3.2 Reviews and Analyses of the Low-Level Tool Requirements

The purpose of these reviews and analyses is to detect and report requirements errors that may have been introduced during the tool design process. These reviews and analyses activities confirm that the low-level tool requirements satisfy these objectives:

a. Compliance with the Tool Requirements: The objective is to ensure that the low-level tool requirements satisfy the Tool Requirements and that derived low-level tool requirements and the design basis for their existence are correctly defined.

b. Accuracy and consistency: The objective is to ensure that each low-level tool requirement is accurate and unambiguous and that the low-level tool requirements do not conflict with each other.

c. Verifiability: The objective is to ensure that each low-level tool requirement can be verified.

d. Conformance to standards: The objective is to ensure that the applicable standards (Tool Design Standards and/or Tool Requirements Standards as defined in the TQP and the Tool Development Plan) were followed during the tool design process, and that deviations from the standards are justified.

e. Traceability: The objective is to ensure that the Tool Requirements and derived tool requirements were developed into the low-level tool requirements.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

f. Algorithm aspects: The objective is to ensure the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities.

6.1.3.3 Reviews and Analyses of the Tool Architecture

The purpose of these reviews and analyses is to detect and report errors that may have been introduced during the development of the tool architecture. These reviews and analyses activities confirm that the tool architecture satisfies these objectives:

a. Compatibility with the Tool Requirements: The objective is to ensure that the tool architecture does not conflict with the Tool Requirements.

b. Consistency: The objective is to ensure that a correct relationship exists between the components of the tool architecture. This relationship exists via data flow and control flow.

c. Conformance to standards: The objective is to ensure that the Tool Design Standards were followed during the tool design process and that deviations from the standards are justified.

d. Protection: The objective is to ensure that if a protection mechanism is used inside the tool architecture, protection breaches are prevented or isolated. See section 11.1 for multi-function tools.

e. External components interface: The objective is to ensure that the interface between the tool and the external components is correctly and completely defined (see 5.2.2.2.g).

6.1.3.4 Reviews and Analyses of the Tool Source Code

The purpose of these reviews and analyses is to detect and report errors that may have been introduced during the tool coding process. These reviews and analyses confirm that the outputs of the tool coding process are accurate, complete, and verifiable. Primary concerns include correctness of the code with respect to the low-level tool requirements and the tool architecture, and conformance to the Tool Code Standards. These reviews and analyses are limited to the Tool Source Code; external component verification is addressed in section 6.1.4.3. These reviews and analyses activities confirm that the Tool Source Code satisfies these objectives:

a. Compliance with the low-level tool requirements: The objective is to ensure that the Tool Source Code is accurate and complete with respect to the low-level tool requirements and that Source Code does not implement any undocumented functions.

b. Compliance with the tool architecture: The objective is to ensure that the Tool Source Code matches the data defined in the tool architecture.

c. Verifiability: The objective is to ensure the Tool Source Code does not contain statements and structures that cannot be verified.

d. Conformance to standards: The objective is to ensure that the Tool Code Standards were followed during the development of the code and that deviations from the standards are justified.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

e. Traceability: The objective is to ensure that the low-level tool requirements were developed into the Tool Source Code.

f. Accuracy and Consistency: The objective is to determine the correctness and consistency of the Tool Source Code, including fixed point arithmetic overflow and resolution, floating-point arithmetic, resource contention and limitations, exception handling, use of uninitialized variables, unused variables or constants, and data corruption due to task conflicts.

6.1.3.5 Reviews and Analyses of the Tool Integration Process Output

The purpose of these reviews and analyses is to detect and report errors that may have been introduced during the tool integration process. The objective is to:

a. Ensure that the outputs of the tool integration process are complete and correct.

Activities include conducting a detailed examination of the compiling and linking data. Typical examples of potential errors include:

- Compiler warnings.
- Missing components.
- Incorrect external components interface.

6.1.4 Tool Testing

The tool testing process is used to demonstrate with a high degree of confidence that the tool satisfies its requirements and that errors that could have an impact on the functionality of the tool have been removed.

6.1.4.1 Tool Testing Objectives

The objectives of tool testing are to execute the Tool Executable Object Code in order to confirm that:

a. The Tool Executable Object Code complies with the Tool Requirements.

b. The Tool Executable Object Code is robust with the Tool Requirements.

c. The Tool Executable Object Code complies with the low-level tool requirements.

d. The Tool Executable Object Code is robust with the low-level tool requirements.

6.1.4.2 Tool Testing Activities

Requirements-based testing is emphasized because this strategy has been found to be the most effective at revealing errors. Activities for requirements-based test case selection include:

a. Test cases and procedures should be developed to demonstrate that the tool satisfies its requirements.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

- Each test case is developed from the requirements (either Tool Requirements or low-level tool requirements) and identifies the set of inputs, the conditions, the expected results, and the pass/fail criteria.

- Test procedures are generated from the test cases.

- Trace Data is generated between the test cases and the test procedures.

b. Normal range tests should be performed, including the following:

- Real and integer input data should be exercised using valid equivalence classes and boundaries values.

- For state transitions, tests cases should be developed to exercise the transitions possible during normal operation.

- For requirements expressed by logic equations or for requirements that include input logic combination, the normal test cases should verify the variable usage and the Boolean operators. In particular, for TQL-1, the tests cases should show that each requirement condition independently affects the required outcome by varying just that requirement condition while holding fixed all other possible requirement conditions.

c. Robustness tests should be performed to address all failure modes (for example, abnormal activation modes, inconsistency inputs, etc.) identified in Tool Requirements.

d. If necessary, additional robustness tests should also be developed to complete the demonstration of the following:

- The ability of the tool to respond to abnormal inputs or conditions.

- The detection of abnormal behavior.

- The prevention of invalid output.

Note: Robustness test cases are requirements-based. The robustness testing criteria cannot be fully satisfied if the tool requirements do not specify the correct response to abnormal conditions and inputs. The test cases may reveal inadequacies in the tool requirements, in which case the tool requirements should be modified. Conversely, if a complete set of requirements exists that covers all abnormal conditions and inputs, the robustness test cases will follow from those tool requirements.

e. Requirements-based test coverage analysis should be performed to demonstrate that all requirements (Tool Requirements and low-level tool requirements) have been tested. This analysis ensures that each requirement has at least one test case and that appropriate normal and robustness testing has been performed.

6.1.4.3 Analysis of Requirements-Based Testing

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

6.1.4.3.1 Objectives

The objective of this analysis with the associated resolution is to provide evidence that the tool's code structure was verified to the degree required for the applicable TQL, and to provide confidence, commensurate with the TQL, in the absence of unintended functionality.

The typical means of accomplishing this is using structural coverage analysis. This analysis may reveal code structure that was not exercised during testing.

Some components of a tool may not be under direct control of the tool developer, such as operating system functions or associated middleware (interface components between the operating system and the application). Typically, this type of component supports the operability of the tool within its operational environment. For functions of external components utilized by the tool, tests should be included that address the interface and functionality of each of those functions, as defined in the Tool Design Description (see 5.2.2.2.g).

However, the following components are typically under control of the tool developer and require structural coverage analysis:

- Components developed for limited use or specifically to support the tool.
- Libraries of functions that are used in tool operation (for example, code that implements basic symbols of a model).

Libraries used or incorporated in the tool's output that become part of the resultant software should be verified as a separate activity from the tool qualification (for example, as part of the software life cycle process).

6.1.4.3.2 Analysis

Activities for analysis of requirements-based testing include:

a. For components under control of the developer of the tool, the analysis should determine which code structure, including interfaces between components, was not exercised by the requirements-based test procedures. For TQL-3, the required level is statement coverage, for TQL-2, the required level is decision coverage, and for TQL-1, the required level is Modified Condition/Decision Coverage (MC/DC). As for any objective, the applicant may propose an alternative means (see 11.5).

b. For components not directly under control of the tool developer (for example, operating system function or external software library), an additional analysis is needed to ensure completion of requirements-based testing. This analysis should ensure that tests exercise the interface and the functionality of each function of the external components utilized by the tool.

c. The analysis should confirm that the requirements-based testing has exercised the data coupling and control coupling between the tool code components.

6.1.4.3.3 Resolution

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

Structural coverage analysis may reveal code structure including interfaces that were not exercised during testing. Resolution would require additional software verification process activity. Possible causes of unexecuted code structure, including interfaces and associated activities to resolve them, include:

- Shortcomings in requirements-based test cases or procedures: The test cases should be supplemented or test procedures changed to provide the missing coverage. The method(s) used to perform the requirements-based coverage analysis may need to be reviewed.

- Inadequacies in requirements: The requirements should be modified, additional test cases developed, and test procedures executed.

- Dead code: The code should either be removed or justified by an impact analysis of the dead code on the tool functionality.

- Deactivated code and unused function: In this case the unexercised Tool Source Code should be either removed or justified. The purpose of this justification is to provide evidence that the unexercised Tool Source Code has no impact on the outputs of the tool in any operational use.

6.1.4.4 Reviews and Analyses of the Test Cases, Procedures, and Results

The purpose of these reviews and analyses is to ensure that the testing of the code was developed and performed accurately and completely. These reviews and analyses activities confirm that the tool testing satisfies these objectives:

a. Test cases: The objective is to ensure that test coverage of all requirements is achieved (see 6.1.4.2.e for associated activities).

b. Test procedures: The objective is to verify that the test cases, including expected results, were correctly developed into test procedures.

c. Test results: The objective is to ensure that the test results are correct and that discrepancies between actual and expected results are explained.

6.1.5 Tool Verification Process Traceability

Tool verification process traceability activities include:

a. Trace Data is developed to show the bi-directional association between Tool Requirements and their associated test cases, as well as between low-level tool requirements and their associated test cases. The purpose of this Trace Data is to support the requirements-based test coverage analysis.

b. Trace Data is developed to show the bi-directional association between test cases and test procedures. The purpose of this Trace Data is to enable verification that the complete set of test cases has been developed into test procedures.

c. Trace Data is developed to show the bi-directional association between test procedures and test results. The purpose of this Trace Data is to enable verification that the complete set of test procedures has been executed.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

6.2 Tool Operational Verification and Validation Process

The tool operational verification and validation process is applied as defined by the tool planning process. The "Tool Operational Verification and Validation" portion of Table T-0 in Annex A summarizes the objectives and outputs of the tool operational verification and validation process by TQL.

6.2.1 Tool Operational Verification and Validation Process Objectives

The purpose of the tool operational verification and validation process is to provide confidence that the outputs and the functionality of the tool comply with the software life cycle process needs. This activity includes both verification and validation.

The verification objectives of the tool operational verification and validation process consist of the detection and reporting of errors that may have been introduced during the tool development processes. Removal of the errors is an activity of the tool development processes. Verification objectives are:

a. Tool Operational Requirements are complete, accurate, verifiable, and consistent.

b. Functionality and the outputs of the tool installed in the tool operational environment comply with the Tool Operational Requirements.

The validation objectives of the tool operational verification and validation process consist of the analysis of the functionality and the outputs of the tools for correctness and completeness with respect to the software life cycle activities performed. Validation objectives are:

aa. Ensure that the Tool Operational Requirements are sufficient and correct to eliminate, reduce, or automate the process(es) identified in the PSAC.

bb. Ensure that the tool meets the needs of the software life cycle process in the tool operational environment.

6.2.2 Tool Operational Verification and Validation Process Activities

The tool operational verification and validation objectives are satisfied through a combination of reviews, analyses, and tests. Reviews, analyses, and development of test cases provide an assessment of the internal consistency and completeness of the Tool Operational Requirements. The execution of the test procedures provides a demonstration of compliance of the Tool Executable Object Code with the Tool Operational Requirements, and its compatibility with the tool operational environment.

The integration of the tool into the tool operational environment should provide demonstration of the correctness and the completeness of the Tool Operational Requirements.

The inputs to the tool operational verification and validation process include the Tool Operational Requirements, the Tool Executable Object Code, and the Tool Verification Plan.

The outputs of the tool operational verification and validation process are Tool Operational Verification Cases and Procedures (see 10.3.3) and the Tool Operational Verification and Validation Results (see 10.3.4).

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

Activities for the tool operational verification and validation activities include:

a. Tool Operational Requirements verification: Reviews and analyses should be performed on the Tool Operational Requirements to ensure that:

   1. Each requirement is accurate, unambiguous, complete, and consistent.

   2. Tool operational environment compatibility requirements are identified.

   3. The expected responses of the tool under abnormal operating conditions are defined, such as abnormal activation modes and inconsistent inputs.

b. Identification and demonstration of relevant process(es): The Tool Operational Requirements should demonstrate the coverage of the process(es) intended to be eliminated, reduced, or automated by the use of the tool.

c. Verification and validation in tool operational environment: The tool installed in the tool operational environment complies with the need of the software life cycle processes. This activity is both the verification of the compliance of the Tool Executable Object Code to the Tool Operational Requirements and the validation of the Tool Operational Requirements. The activities are:

   1. Test cases and procedures should be developed to ensure that the tool satisfies the Tool Operational Requirements. These test cases and procedures should be requirements-based.

   2. Test procedures should be executed after the installation of the Tool Executable Object Code in the tool operational environment.

   3. Test results should be reviewed to ensure that test results are correct and that discrepancies between actual and expected results are explained.

   4. In order to validate the Tool Operational Requirements the tool should be exercised. The set of inputs selected should represent the actual tool use and its interfaces in the tool operational environment. This validation should confirm that the Tool Operational Requirements are correct and complete.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

7.0 TOOL CONFIGURATION MANAGEMENT PROCESS

This section discusses the objectives and activities of the tool configuration management (TCM) process. The TCM process is applied as defined by the tool planning process (see 4) and the Tool Configuration Management Plan (see 10.1.5). Outputs of the TCM process are recorded in Tool Configuration Management Records (see 10.1.13) or in other tool life cycle data.

The TCM process, working in cooperation with the other tool life cycle processes, assists in:

a. Providing a defined and controlled configuration of the tool throughout the tool life cycle.

b. Providing the ability to consistently replicate the Tool Executable Object Code or to regenerate it in case of a need for investigation or modification.

c. Providing control of process inputs and outputs during the tool life cycle that ensures consistency and repeatability of process activities.

d. Providing a known point for review, assessing status, and change control by control of configuration items and the establishment of baselines.

e. Providing controls that ensure problems receive attention and changes are recorded, approved, and implemented.

f. Providing evidence of approval of the tool by control of the outputs of the tool life cycle processes.

g. Assessing the tool product compliance with requirements.

h. Ensuring that secure physical archiving, recovery, and control are maintained for the configuration items.

7.1 Tool Configuration Management Process Objectives

The TCM process objectives are:

a. Each configuration item and its successive versions are labeled unambiguously so that a basis is established for the control and reference of configuration items.

b. Baselines are defined for further tool life cycle process activity and allow reference to, control of, and traceability between configuration items.

c. The problem reporting process records process non-compliance with tool plans and standards, records deficiencies of outputs of tool life cycle processes, records anomalous behavior of tool products, and ensures resolution of these problems.

d. Change control provides for recording, evaluation, resolution, and approval of changes throughout the tool life cycle.

e. Change review ensures problems and changes are assessed, problems and changes are approved or disapproved, approved changes are implemented, and feedback is provided to affected processes through problem reporting and change control methods defined during the tool planning process.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

f. Status accounting provides data for the configuration management of tool life cycle processes with respect to configuration identification, baselines, problem reports, and change control.

g. Archival and retrieval ensures that the tool life cycle data associated with the tool product can be retrieved in case of a need to duplicate, regenerate, retest, or modify the tool product. The objective of the release activity is to ensure that only the authorized tool is used in the software life cycle processes, in addition to being archived and retrievable.

h. Tool life cycle environment control ensures that other tools used to produce the tool itself are identified, controlled, and retrievable.

The objectives for TCM are independent of TQL. However, two categories of tool life cycle data may exist based on the TCM controls applied to the data (see 7.3).

Table T-8 of Annex A is a summary of the objectives and outputs of the TCM process by TQL.

7.2 Tool Configuration Management Process Activities

The TCM process includes the activities of configuration identification, change control, baseline establishment, and archiving of the tool product, including the related tool life cycle data. The TCM process does not stop when the tool product is accepted by the certification authority, but continues throughout the service life of the system or equipment of the resultant software. If tool life cycle activities will be performed by a supplier, then configuration management activities should be applied to the supplier.

7.2.1 Configuration Identification

Activities include:

a. Configuration identification should be established for the tool life cycle data.

b. Configuration identification should be established for each configuration item, for each separately controlled component of a configuration item, and for combinations of configuration items that comprise a tool product.

c. Configuration items should be configuration identified prior to the implementation of change control and traceability analysis.

d. A configuration item should be configuration identified before that item is used by other tool life cycle processes or referenced by other life cycle data.

7.2.2 Baselines and Traceability

Activities include:

a. Baselines should be established for configuration items used for certification credit. (Intermediate baselines may be established to aid in controlling tool life cycle process activities.)

b. A tool product baseline should be established for the tool product and defined in the Tool Configuration Index (see 10.1.11).

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. Not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

c. Baselines should be established in controlled software libraries (physical, electronic, or other) to ensure their integrity. Once a baseline is established, it should be protected from change.

d. Change control activities should be followed to develop a derivative baseline from an established baseline.

e. A baseline should be traceable to the baseline from which it was derived, if a credit is sought for tool life cycle process activities or data associated with the development of the previous baseline.

f. A configuration item should be traceable to the configuration item from which it was derived, if a credit is sought for tool life cycle process activities or data associated with the development of the previous configuration item.

g. A baseline or configuration item should be traceable either to the output it identifies or to the process with which it is associated.

7.2.3 Problem Reporting, Tracking, and Corrective Action

Activities include:

a. A Tool Problem Report should be prepared that describes the process non-compliance with plans, output deficiency, or tool anomalous behavior, and the corrective action taken, as defined in section 10.1.12.

Note: Separate problem reporting systems may be used. For example, errors detected during the tool operational verification and validation process may be recorded in another problem reporting system than errors detected during the tool verification process.

b. Problem reporting should provide for configuration identification of affected configuration item(s) or definition of affected process activities, status reporting of Tool Problem Reports, and approval and closure of Tool Problem Reports.

c. Tool Problem Reports that require corrective action of the tool product or outputs of tool life cycle processes should invoke the change control activity.

7.2.4 Change Control

Activities include:

a. Change control should preserve the integrity of the configuration items and baselines by providing protection against their change.

b. Change control should ensure that any change to a configuration item requires a change to its configuration identification.

c. Changes to baselines and to configuration items under change control to produce derivative baselines should be recorded, approved, and tracked. Problem reporting is related to change control, since resolution of a reported problem may result in changes to configuration items or baselines.

Note: It is generally recognized that early implementation of change control assists the control and management of tool life cycle process activities.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

d. Tool changes should be traced to their origin and the tool life cycle processes repeated from the point at which the change affects their outputs. For example, an error that is shown to result from an incorrect design should result in design correction, code correction, and repetition of the associated integral process activities.

e. Throughout the change activity, tool life cycle data affected by the change should be updated and records should be maintained for the change control activity.

The change control activity is aided by the change review activity.

7.2.5 Change Review

Activities include:

a. Confirming that all affected configuration items are identified.

b. Assessing the impact of the change on Tool Operational Requirements.

c. Assessing the problem or change, with decisions for action to be taken.

d. Providing feedback of Tool Problem Report or change impact and decisions to affected processes.

e. Assessing the software process feedback on tool problems and proposed changes identified by the tool process.

7.2.6 Configuration Status Accounting

Activities include:

a. Reporting on configuration item identification, baseline identification, Tool Problem Report status, change history, and release status.

b. Defining the data to be maintained and the means to record and report the data's status.

7.2.7 Archive, Retrieval, and Release

Activities include:

a. Tool life cycle data associated with the tool product should be retrievable from the approved source (for example, an archive at the developing organization or company).

b. Procedures should be established to ensure the integrity of the stored data (regardless of medium of storage) by:

1. Ensuring that no unauthorized changes can be made.

2. Selecting storage media that minimize regeneration errors or deterioration.

3. Preventing loss or corruption of data over time. Depending on the storage media used, this may include periodically exercising the media or refreshing the archived data.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. Not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

4. Storing duplicate copies in physically separate archives that minimize the risk of loss in the event of a disaster.

c. The duplication process should be verified to produce accurate copies, and procedures should exist that ensure error-free copying of the Tool Executable Object Code.

d. Configuration items should be identified and released prior to use in the scope of the software life cycle processes.

e. Data retention procedures should be established to enable tool modifications.

Note: Additional data retention considerations may include items such as business needs and future certification authority reviews, which are outside the scope of this document.

7.3 Data Control Categories

Tool life cycle data can be assigned to one of two categories: Control Category 1 (CC1) and Control Category 2 (CC2). These categories are related to the configuration management controls placed on the data. Table 7-1 defines the set of TCM process activities associated with each control category, where● indicates that the activities apply for tool life cycle data of that category. CC2 activities are a subset of the CC1 activities.

The tables of Annex A specify the control category for each tool life cycle data item, by TQL. Guidance for data control categories includes:

a. The TCM process activities for tool life cycle data categorized as CC1 should be applied according to Table 7-1.

b. The TCM process activities for tool life cycle data categorized as CC2 should be applied according to Table 7-1, as a minimum.

7.4 Tool Life Cycle Environment Control

The tool life cycle environment is defined by the tool planning process and identified in the Tool Life Cycle Environment Configuration Index (see 10.1.10). Activities include:

a. Configuration identification should be established for the Executable Object Code (or equivalent) of the tools used to develop, control, build, and to verify the tool product and its data.

b. The TCM process for controlling the tools used to produce the Tool Executable Object Code (or equivalent) (for example, compilers, assemblers, linkage editors) should comply with the objectives associated with CC2 data, as a minimum.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. ©2011 RTCA, Inc.

Table 7-1 TCM Process Activites Associated with CC1 and CC2 Data

TCM Process Activity	Reference	CC1	CC2
Configuration Identification	7.2.1	●	●
Baselines	7.2.2.a 7.2.2.b 7.2.2.c 7.2.2.d 7.2.2.e	●	
Traceability	7.2.2.f 7.2.2.g	●	●
Problem Reporting	7.2.3	●	
Change Control – integrity and identification	7.2.4.a 7.2.4.b	●	●
Change Control – tracking	7.2.4.c 7.2.4.d 7.2.4.e	●	
Change Review	7.2.5	●	
Configuration Status Accounting	7.2.6	●	
Retrieval	7.2.7.a	●	●
Protection against Unauthorized Changes	7.2.7.b.1	●	●
Media Selection, Refreshing, Duplication	7.2.7.b.2 7.2.7.b.3 7.2.7.b.4 7.2.7.c	●	
Release	7.2.7.d	●	
Data Retention	7.2.7.e	●	●

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. Do not for sale, transfer, or distribution to any other party. See Electronic L

8.0 TOOL QUALITY ASSURANCE PROCESS

This section discusses the objectives and activities of the tool quality assurance (TQA) process. The TQA process is applied as defined by the tool planning process (section 4) and the Tool Quality Assurance Plan (see 10.1.6). Outputs of the TQA process activities are recorded in Tool Quality Assurance Records (see 10.1.14) or other tool life cycle data.

The TQA process assesses the tool life cycle processes and their outputs to obtain assurance that:

a. Objectives are satisfied.

b. Deficiencies are detected, evaluated, tracked, and resolved.

c. The tool product and tool life cycle data conform to this document and additional regulatory requirements, if any.

8.1 Tool Quality Assurance Process Objectives

The TQA process objectives provide confidence that the tool life cycle processes produce a tool that conforms to its requirements by assuring that these processes are performed in compliance with the approved tool plans and standards.

The objectives of the TQA process are to:

a. Obtain assurance that tool development plans and standards are developed and reviewed for consistency.

b. Obtain assurance that tool development processes and integral processes, including those of suppliers, comply with approved tool plans and standards.

c. Obtain assurance that the transition criteria for the tool life cycle processes are satisfied.

d. Conduct a conformity review of the tool product.

The applicability of the objectives by TQL is specified in Table T-9 of Annex A.

8.2 Tool Quality Assurance Process Activities

Activities for satisfying the TQA process objectives include:

a. Those performing the TQA process should take an active role in the activities of the tool life cycle processes, and have the authority, responsibility, and independence to ensure that the TQA process objectives are satisfied.

b. The TQA process should provide assurance that tool plans and standards are developed and reviewed for consistency.

c. The TQA process should provide assurance that the tool life cycle processes comply with the approved tool plans and standards.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

d. The TQA process should include audits of the tool development and integral processes during the tool life cycle to obtain assurance that:

1. Tool plans are available as specified in section 4.

2. Deviations from the tool plans and standards are detected, recorded, evaluated, tracked, and resolved.

Note: It is generally accepted that early detection of process deviations assists efficient achievement of tool life cycle process objectives.

3. Approved deviations are recorded.

4. The tool development environment has been provided as specified in the tool plans.

5. The problem reporting, tracking, and corrective action process complies with the Tool Configuration Management Plan.

Note: Monitoring of the activities of tool life cycle processes may be performed to provide assurance that the activities are under control.

e. The TQA process should provide assurance that the transition criteria for the tool life cycle processes have been satisfied in compliance with the approved tool plans.

f. The TQA process should provide assurance that tool life cycle data is controlled in accordance with the control categories as defined in section 7.3 and the tables of Annex A.

g. A tool conformity review should be conducted prior to the delivery of the tool product which is intended to be used by resultant software submitted as part of a certification application.

h. The TQA process should produce records of the TQA process activities (section 10.1.14), including audit results and evidence of completion of the tool conformity review for each tool product.

i. The TQA process should provide assurance that any supplier processes and outputs comply with approved tool plans and standards.

8.3 Tool Conformity Review

The purpose of the tool conformity review is to obtain assurances, for a tool product, that the tool life cycle processes are complete, tool life cycle data is complete, and the Tool Executable Object Code is controlled and can be regenerated.

The tool conformity review may be supplemented or performed in the context of a software conformity review.

This review should determine that:

a. Planned tool life cycle process activities, including the generation of tool life cycle data, have been completed and records of their completion are retained.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. Not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

b. Evidence exists that tool life cycle data have been produced in accordance with tool plans and standards, and is controlled in accordance with the TCM Plan.

c. Evidence exists that Tool Problem Reports have been evaluated and have their status recorded.

d. Tool requirement deviations are recorded and approved.

e. The Tool Executable Object Code can be regenerated from the archived Tool Source Code.

f. Tool Problem Reports deferred from a previous tool conformity review are re-evaluated to determine their status.

g. If certification credit is sought for the use of previously developed tools, the current tool product baseline is traceable to the previous baseline and the approved changes to that baseline.

Note: For post-qualification tool modifications, a subset of the tool conformity review activities, as justified by the significance of the change, may be performed.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

This Page Intentionally Left Blank

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

9.0 TOOL QUALIFICATION LIAISON PROCESS

The tool qualification liaison process is not performed alone, but in the framework of the certification liaison process (or equivalent, depending on the domain) of the software. This section identifies the tool qualification aspects of this process.

Liaison with the certification authority is needed for tool qualification. The PSAC identifies the need for tool qualification according to the specific use of the tool in the software life cycle process.

In a software life cycle process, if the PSAC identifies that one or more tools need to be qualified, the qualification liaison process includes the following activities:

a. The primary input submitted to the certification authority is the PSAC containing the tool qualification identification. The minimum data that should be submitted to the certification authority is:

- Plan for Software Aspects of Certification (PSAC).
- Tool Qualification Plan (TQP).
- Tool Configuration Index (TCI).
- Tool Accomplishment Summary (TAS).
- Software Life Cycle Environment Configuration Index (SECI).
- Software Accomplishment Summary (SAS).

For TQL-5, the TQP, TCI, and TAS are not required.

Other data or evidence of compliance from the tool life cycle may be submitted as requested by the certification authority. All data should be available for certification authority review.

b. Both the PSAC and TQP should be submitted to the certification authority in order to obtain agreement on the means of compliance.

c. As evidence that the tool life cycle processes satisfy the plans, the tool life cycle data should be available to the certification authority for review.

d. All the known problems and functional limitations should be analyzed from the software life cycle process perspective, and a report should be provided in the Software Accomplishment Summary. The impact of known problems and functional limitations of the tool should be identified and analyzed to ensure that the functionality and the outputs of the tool comply with the Tool Operational Requirements.

e. The Software Life Cycle Environment Configuration Index should provide the identification of all the tools used in the scope of the applicable software release. For TQL-5, it should also include references to the tool qualification data (Tool Operational Requirements and tool operational verification and validation data). For

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

other TQLs, the Software Life Cycle Environment Configuration Index should reference the TCI.

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

10.0 TOOL QUALIFICATION DATA

The packaging of the tool qualification data is at the discretion of the applicant. For example, data items may be combined into a common document or divided over multiple documents.

This section provides guidance for the typical tool qualification life cycle data produced and the minimum contents of each data item. This section is divided into three subsections pertinent to tool qualification life cycle data:

- Tool qualification liaison process and other integral processes data (see 10.1).
- Tool qualification data produced during the tool development process and corresponding verification activities (see 10.2).
- Tool qualification data produced during the Tool Operational Requirements process and during the tool operational integration process and corresponding verification and validation activities (see 10.3).

10.1 Tool Qualification Liaison Process and Other Integral Processes Data

10.1.1 Tool-Specific Information in PSAC

The PSAC should include the following information about the need for tool qualification:

a. Identification of the tool and its intended use, including its impact in the software life cycle process.

b. Details of the certification credit sought through tool use for eliminating, reducing, or automating the process(es).

c. Substantiation of the maturity and technical background of any technology or theory (for example, mathematical theory) implemented in the tool to show its applicability.

d. The TQL proposed for the tool and supporting justification.

e. Tool source (for example, in-house, COTS, third party).

f. The stakeholders involved in the tool qualification and their specific roles and responsibilities, including who is responsible for satisfying specific objectives.

g. Description of the Tool Operational Requirements definition process (see 5.1), tool operation integration process (see 5.3), and tool operational V&V process (see 6.2) (or a reference to where these processes will be described).

h. Description of the tool operational environment in which the tool will be used.

i. If the tool qualification data is reused, identify previous applications of the tool, the strategy and justification for reuse, and any applicable re-qualification activities. In the case of a third party tool or a COTS tool, information about previous application can be provided by the supplier since it may not be available from the users of the tool. See sections 11.2 and 11.3 for reuse of previously qualified tools and COTS tools.

For Personal Use by Jordan Colson, Skyryse © 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

j. Reference to the TQP, or if no TQP is generated (for TQL-5), reference the data to support tool qualification.

10.1.2 Tool Qualification Plan (TQP)

For a tool to be qualified at TQL 1, 2, 3 or 4, the TQP should describe the tool qualification process. This plan should include:

a. Identification of the tool, and, if applicable, user configuration.

b. Qualification considerations, including the proposed TQL and means of compliance with the objectives of this document.

c. A functional overview of the tool, its interfaces, and its architecture. Additionally, any external components should be identified.

d. Description of the tool operational environment(s), and if different, the tool verification environment(s).

e. Description of the means the applicant will use to provide the certification authority with visibility of the activities of the tool life cycle processes so tool reviews can be planned.

f. Tool life cycle description and the qualification activities to be performed.

g. Tool qualification data to be produced.

h. Any additional considerations that may affect the qualification process, for example, deactivated code, COTS tools, reuse, tool qualification (of other tools used to develop or verify the tool), alternate means of qualification, tool service history, and means to ensure the determinism of the tool per the last paragraph of section 2.0 of this document.

i. Organizational responsibilities within the tool life cycle processes.

j. If suppliers are used, a means of ensuring that supplier processes and outputs will comply with approved tool plans and standards.

10.1.3 Tool Development Plan

The Tool Development Plan includes the objectives, standards, and tool life cycle(s) to be used in the tool development processes. This plan should include:

a. Standards: Identification of the Tool Requirements Standards, Tool Design Standards, and Tool Code Standards for the project.

b. Tool life cycle: A description of the tool life cycle processes to be used to form the specific tool life cycle(s) to be used on the project, including the transition criteria for the tool development processes. This description is distinct from the summary provided in the TQP, in that it provides the development activity description with enough detail necessary to ensure proper implementation of the tool life cycle processes.

c. Tool development environment: A statement of the chosen tool development environment in terms of hardware and software, including:

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

1. The chosen requirements development method(s) and tools to be used.

2. The chosen design method(s) and tools to be used.

3. The chosen coding method(s), programming language(s), and coding tool(s) to be used.

4. The compilers and linkage editors to be used.

5. The selected workstation and operating system identification.

10.1.4 Tool Verification Plan

The Tool Verification Plan is a description of the verification activities to satisfy the tool verification process objectives. These procedures may vary by TQL as defined in Annex A tables. This plan should include:

a. Independence: A description of the methods for establishing verification independence, when required.

b. Verification methods: A description of the verification methods to be used for each activity of the tool verification process, including the following:

   1. Review methods, including checklists or other aids.

   2. Analysis methods, including traceability and coverage analysis.

   3. Testing methods, including the method for selecting test cases, the test procedures to be used, and the test data to be produced.

c. Transition criteria: The transition criteria for entering the tool verification process defined in the plan.

d. Protection considerations: If protection is used, the methods used to verify the integrity of the protection.

e. Reverification method: For modification, a description of the methods for identifying, analyzing, and verifying the affected areas of the tool.

f. Previously developed components in the tool product: For these components, if the initial compliance baseline for the verification process does not comply with this document, a description of the methods to satisfy the objectives of this document.

For Personal Use by Jordan Colson, Skyryse © 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

10.1.5 Tool Configuration Management Plan

The Tool Configuration Management Plan establishes the methods to be used to satisfy the objectives of the TCM process throughout the tool life cycle. This plan should include:

a. Environment: A description of the TCM environment to be used, including procedures, tools, methods, standards, organizational responsibilities, and interfaces.

b. Activities: A description of the TCM process activities in the tool life cycle, including:

1. Configuration identification: Items to be identified, when they will be identified, and identification methods for tool life cycle data.

2. Baselines and traceability: The means of establishing baselines, what baselines will be established, when these baselines will be established, the software library controls, and the configuration item and baseline traceability.

3. Problem reporting: The content and identification of Tool Problem Reports for the tool and tool life cycle processes, when they will be written, the method of closing Tool Problem Reports, and the relationship to the change control activity.

4. Change control: Configuration items and baselines to be controlled, when they will be controlled, the problem/change control activities that control them, pre-qualification controls, post-qualification controls, and the means of preserving the integrity of baselines and configuration items.

5. Change review: The method of handling feedback to and from the tool life cycle processes; the methods of assessing and prioritizing problems, approving changes, and handling their resolution or change implementation; and the relationship of these methods to the problem reporting and change control activities.

6. Configuration status accounting: The data to be recorded to enable reporting configuration management status, definition of where that data will be kept, how it will be retrieved for reporting, and when it will be available.

7. Archive, retrieval, and release: The integrity controls, the release method and authority, and data retention.

8. Tool life cycle environment controls: Controls for the tools used to develop, build, verify, and load the tool addressing items 1 through 7 above.

9. Tool life cycle data controls: Controls associated with CC1 and CC2 data.

c. Transition criteria: The transition criteria for entering the TCM process.

d. TCM data: A definition of the tool life cycle data produced by the TCM process, including TCM Records, the Tool Configuration Index, and the Tool Life Cycle Environment Configuration Index.

e. Supplier control: The means of applying TCM process requirements to suppliers.

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

10.1.6 Tool Quality Assurance Plan

The TQA Plan establishes the methods to be used to satisfy the objectives of the TQA process. The TQA Plan may include descriptions of process improvement, metrics, and progressive management methods. This plan should include:

a. Environment: A description of the TQA environment, including scope, organizational responsibilities and interfaces, standards, procedures, tools, and methods.

b. Authority: A statement of the TQA authority, responsibility, and independence.

c. Activities: The TQA activities that are to be performed for each tool life cycle process and throughout the tool life cycle including:

   1. TQA methods, for example, reviews, audits, reporting, inspections, and monitoring of the tool life cycle processes.

   2. Activities related to the problem reporting, tracking, and corrective action system.

   3. A description of the tool conformity review activity.

d. Transition criteria: The transition criteria for entering the TQA process.

e. Timing: The timing of the TQA process activities in relation to the activities of the tool life cycle processes.

f. TQA Records: A definition of the records to be produced by the TQA process.

g. Supplier oversight: A description of the means of ensuring that supplier's processes and outputs will comply with the approved tool plans and standards.

10.1.7 Tool Requirements Standards

The purpose of Tool Requirements Standards is to define the methods, rules, and tools to be used to develop the Tool Requirements. These standards should include:

a. The methods to be used for developing Tool Requirements, such as structured methods.

b. Notations to be used to express requirements, such as data flow diagrams and formal specification languages.

c. Constraints on tools used to develop Tool Requirements.

d. The method to be used to analyze the effect of derived tool requirements on the Tool Operational Requirements (see 5.2.1.2.h).

10.1.8 Tool Design Standards

The purpose of Tool Design Standards is to define the methods, rules, and tools to be used to develop the tool architecture and low-level tool requirements. These standards should include:

a. Design description method(s) to be used.

For Personal Use by Jordan Colson, Skyryse © 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

b. Naming conventions to be used.

c. Constraints on tools used to develop Tool Design Description.

d. Complexity restrictions, for example, maximum level of nested calls or conditional structures, use of unconditional branches, and number of entry and exit points of code components.

Note: In some cases the Tool Requirements Standards may be used for the low-level tool requirements rather than the Tool Design Standards.

10.1.9 Tool Code Standards

The purpose of the Tool Code Standards is to define the programming languages, methods, rules, and tools to be used to produce the Tool Source Code. These standards should include:

a. Programming language(s) to be used and/or defined subset(s). For a programming language, reference the data that unambiguously defines the syntax, the control behavior, the data behavior, and side-effects of the language. This may require limiting the use of some features of a language.

b. Tool Source Code presentation standards (for example, line length restriction, indentation, and blank line usage) and Tool Source Code documentation standards (for example, name of author, revision history, inputs and outputs, and affected global data).

c. Naming conventions for components, subprograms, variables, and constants.

d. Conditions and constraints imposed on permitted coding conventions, such as the degree of coupling between software components and the complexity of logical or numerical expressions and rationale for their use.

e. Constraints on tools used to develop Tool Source Code.

10.1.10 Tool Life Cycle Environment Configuration Index

The Tool Life Cycle Environment Configuration Index identifies the configuration of the tool life cycle environment. This index is written to aid reproduction of the tool development and verification environment, for regeneration of executable code, and for tool reverification or modification. The index should:

a. Identify the tool development environment, such as workstation and operating system.

b. Identify the tools used to develop the tool submitted for qualification (such as, compilers and linkage editors).

c. Identify the tool verification environment(s), if different from the tool development environment.

d. Identify qualified tools and their associated tool qualification data. These tools are those used in the context of the life cycle processes of the tool submitted for qualification.

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

10.1.11 Tool Configuration Index

The Tool Configuration Index identifies the configuration of the tool product. Specific configuration identifiers and version identifiers should be provided.

The Tool Configuration Index should identify:

a. The tool product.

b. Tool Executable Object Code, including all associated files.

c. Each Tool Source Code component.

d. Previously developed components in the tool product, if used.

e. Tool life cycle data.

f. Reference to the Tool Life Cycle Environment Configuration Index (see 10.1.10), if it is packaged separately.

g. Archive and release media.

h. Instructions for building the Tool Executable Object Code (including items such as instructions and data for compiling and linking) and the procedures used to recover the tool for regeneration, testing, or modification.

Note 1: The Tool Configuration Index can contain one data item or a set (hierarchy) of data items. The Tool Configuration Index can contain the items listed above or it may reference another Tool Configuration Index or other identified configuration data that specifies the individual items and their versions.

Note 2: The Tool Configuration Index may be produced for one tool product version or it may be extended to contain data for several alternative or successive tool product versions.

Note 3: The Tool Configuration Index may include only the data produced during the tool development process. It may not identify all the data of the tool integrated in the tool operational environment. In this case, the Software Life Cycle Environment Configuration Index for the software using the tool should identify the data.

10.1.12 Tool Problem Reports

Tool Problem Reports are a means to identify and record the resolution to tool product anomalous behavior, process non-compliance with tool plans and standards, and deficiencies in tool life cycle data. Tool Problem Reports should include:

a. Identification of the configuration item and/or the tool life cycle process activity in which the problem was observed.

b. Identification of the configuration item(s) to be modified or a description of the process to be changed.

c. A problem description that enables the problem to be understood and resolved.

For Personal Use by Jordan Colson, Skyryse © 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

d. A description of the corrective action taken to resolve the reported problem.

10.1.13 Tool Configuration Management Records

The results of the TCM process activities are recorded in TCM Records. Examples include configuration identification lists, baseline, change history reports, archive records, and release records. These examples do not imply the need to produce records of these specific types.

Note: Due to the integral nature of the TCM process, its outputs will often be included as parts of other tool life cycle data.

10.1.14 Tool Quality Assurance Records

The results of the TQA process activities are recorded in TQA Records. These may include TQA review or audit reports, meeting minutes, records of authorized process deviations, and tool conformity review records.

In addition to the TQA Records which address the development of the tool, there will be Software Quality Assurance Records which address its use.

10.1.15 Tool Accomplishment Summary

For a tool to be qualified at TQL 1, 2, 3, or 4, the Tool Accomplishment Summary is the primary data item for showing compliance with the TQP.

The Tool Accomplishment Summary should include:

a. Tool overview: This section briefly describes the tool's main functions, its inputs and output, and any differences from the tool overview proposed in the TQP.

b. Qualification considerations: This section restates the TQL proposed for the tool described in the TQP. It also identifies the operational environment(s).

c. Tool life cycle: This section summarizes the actual tool life cycle(s) and explains differences from the tool qualification activities identified in the TQP.

d. Tool life cycle data: This section describes any differences from the proposals made in the TQP for the tool life cycle data produced, the relationship of the data to each other and to other data defining the tool, and the means by which the data was made available to the certification authority. This section explicitly references, by configuration identifiers and version, the applicable Tool Configuration Index and Tool Life Cycle Environment Configuration Index. Detailed information regarding configuration identifiers and specific versions of tool life cycle data is provided in the Tool Configuration Index.

e. Additional considerations: This section summarizes qualification issues that may warrant the attention of the certification authority. It explains any differences from the proposals contained in the TQP regarding such considerations. References should be made to the data items applicable, such as special conditions or interpretive material.

f. Supplier Oversight: This section describes how supplier processes and outputs comply with approved tool plans and standards.

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

g. Tool identification: This section identifies the tool configuration by part number and version.

h. Change history: If applicable, this section includes a summary of changes affecting tool functionality and identifies any changes from or improvements to the tool life cycle process since the previous qualification.

i. Tool status: This section contains a summary of Tool Problem Reports unresolved at the time of approval and an assessment of the problem's impact on tool functionality. Additionally, it should include any functional limitations of the tool.

j. Compliance statement: This section includes a statement of compliance with this document and a summary of the methods used to demonstrate compliance with criteria specified in the tool plans. This section also addresses additional rulings made by the certification authority and any deviations from the tool plans, standards, and this document not covered elsewhere in the Tool Accomplishment Summary.

10.1.16 Tool-Specific Information in Software Accomplishment Summary

For each tool to be qualified at TQL 1, 2, 3, or 4, the SAS should include:

a. Identification of the tool.

b. Details of the certification credit sought through tool use, that is, the software process(es) to be eliminated, reduced, or automated.

c. Reference to the Tool Accomplishment Summary.

d. Compliance statement of the tool development, verification, and integral processes to the tool plans, including tool user activities.

e. Analysis of open Tool Problem Reports to ensure that the behavior of the tool still complies with the Tool Operational Requirements.

f. Description of any tool use differences as identified in the PSAC.

For tools to be qualified at TQL-5, the SAS should provide the following information:

aa. Identification of the tool.

bb. Details of the certification credit sought through tool use, that is, the software process(es) to be eliminated, reduced, or automated.

cc. Reference to the tool qualification data.

10.1.17 Tool-Specific Information in Software Life Cycle Environment Configuration Index (SECI)

For each tool to be qualified at TQL 1, 2, 3, or 4, the SECI should include:

a. Identification of the tool.

For Personal Use by Jordan Colson, Skyryse © 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

b. Reference to the Tool Configuration Index and TAS.

For tools to be qualified at TQL-5, the Software Life Cycle Environment Configuration Index should include:

aa. Identification of the tool.

bb. Reference to the tool qualification data.

10.2 Tool Qualification Data Produced During the Tool Development Processes and Corresponding Verification Activities

10.2.1 Tool Requirements

The Tool Requirements describe all the tool functionality. This data should include:

a. A description of the tool functions and technical features, including modes of operation.

b. User instructions, installation instructions, list of error messages, and constraints. This is often packaged as a user manual. The user manual may be part of the Tool Requirements or it may be packaged in one or more documents.

c. Requirements to describe the user's ability to customize the tool.

d. Functional requirements, with the appropriate level of detail (see 5.2.1.2.k).

e. Specific requirements, if necessary, for compliance with tool operational environment.

f. Specific requirements to address the failure modes and response to inconsistent inputs.

g. The expected responses of the tool under abnormal operating conditions.

h. For a collection of tools, interface requirements between the tools within the collection.

10.2.2 Tool Design Description

The Tool Design Description is a definition of the tool architecture and low-level tool requirements that will refine the Tool Requirements. This data should include:

a. The description of the tool architecture defining the tool structure to implement the requirements.

b. A detailed description of how the Tool Requirements are all allocated into the tool architecture.

c. The input/output description (for example, a data dictionary) both internally and externally throughout the software architecture.

d. The data flow and control flow of the design.

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

e. Scheduling procedures.

f. Protection, if used.

g. Descriptions of the software components, whether they are new or previously developed, and, if previously developed, reference to the baseline from which they were taken.

h. The low-level tool requirements which implement and are traceable to the Tool Requirements.

i. The derived low-level tool requirements resulting from the tool design process.

10.2.3 Tool Source Code

For each Tool Source Code component, this data consists of: (1) code written in script or source language(s); (2) linking instructions; and (3) the compiler instructions for generating the Tool Executable Object Code from the Tool Source Code (if used, which depends on the language). This data should include the tool identification, including the name and date of revision and/or version, as applicable.

10.2.4 Tool Executable Object Code

The Tool Executable Object Code consists of a form of the tool that is directly usable by the processing unit of the tool operational environment. It should be noted that depending on the selected source code language, compiling and linking may not be necessary. One example is a tool implemented in a script language. In this case, the Tool Executable Object Code should be understood to be the tool in its executable format.

Associated files, such as parameter and configuration files required to execute the tool, are considered to be included in the Tool Executable Object Code. The Tool Executable Object Code is generated in the tool development environment. Several Tool Executable Object Code configurations may be generated if a tool is intended to be used in multiple tool operational environments.

10.2.5 Tool Verification Cases and Procedures

The Tool Verification Cases and Procedures detail how the tool verification process activities are implemented. This data should include:

a. Review and analysis procedures: The scope and depth of the review or analysis methods to be used, in addition to the description in the Tool Verification Plan.

b. Test cases: The purpose of each test case, set of inputs, conditions, expected results to achieve the required coverage criteria, and the pass/fail criteria.

c. Test procedures: The step-by-step instructions for how each test case is to be set up and executed, how the test results are evaluated, and the test environment to be used.

10.2.6 Tool Verification Results

The Tool Verification Results are produced by the tool verification process activities, and should:

For Personal Use by Jordan Colson, Skyryse © 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

a. For each review, analysis, and test, indicate each procedure that passed or failed during the activities and the final pass/fail results.

b. Identify the configuration item or software version reviewed, analyzed, or tested.

c. Include the results of tests, reviews, and analyses, including coverage analyses and traceability analyses.

d. Record and track any discrepancies found using the problem reporting process.

Note: This data does not include the data produced by the tool operational verification and validation process (these are addressed in sections 10.3.3 and 10.3.4 of this document).

10.2.7 Trace Data

Trace Data establishes the associations between life cycle data items contents. Trace Data should be provided that demonstrates bi-directional associations between:

a. Tool Operational Requirements and Tool Requirements.

b. Tool Requirements and low-level tool requirements.

c. Low-level tool requirements and Tool Source Code.

d. Tool Requirements and low-level tool requirements, and their associated test cases.

e. Test cases and test procedures.

f. Test procedures and test results.

Note: The Trace Data between test procedures and test results may be part of the Tool Verification Results (see 10.2.6.c).

10.3 Tool Qualification Data Produced During the Tool Operational Requirements Process, the Tool Operational Integration Process, and Corresponding Verification and Validation Activities

10.3.1 Tool Operational Requirements

The Tool Operational Requirements define the tool's functionality and interface from a software life cycle process perspective (that is, the process which uses the tool). The Tool Operational Requirements should include, as applicable:

a. Description of the context of the tool use, including interfaces with other tools and integration of the tool output files into the resultant software.

b. Description of the tool operational environment(s) (where the tool will be installed).

c. Description of input files, including format, language definition, etc.

d. Description of output files, including format and contents.

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

e. Requirements for all the tool functions and technical features used to satisfy the identified software life cycle process(es).

f. Requirements to address the abnormal activation modes or inconsistency inputs that should be detected by the tool. These requirements should consider the impact of those modes on the functionality and outputs of the tool. (This item is not applicable to TQL-5.)

g. The applicable user information, such as a user manual and installation guide or a reference to it, if not provided as part of the Tool Requirements data.

h. Description of the operational use of the tool (including selected options, parameters values, command line, etc.).

i. Performance requirements specifying the behavior of the tool output.

10.3.2 Tool Installation Report

The Tool Executable Object Code is installed into the tool operational environment. A report of this installation should include:

a. Identification of the tool operational environment (hardware configuration, operating system configuration, installed options, etc.).

b. Identification of the version of the Tool Executable Object Code, including all configuration and parameters files.

c. Identification, if necessary, of the external components.

d. Identification of the means to execute the tool (command line, scripts files, etc.); this could include a reference to the user manual.

Note: This information may be included in the SECI.

10.3.3 Tool Operational Verification and Validation Cases and Procedures

The Tool Operational Verification and Validation Cases and Procedures detail how the tool operational verification and validation process activities are performed. This data should include:

a. Review and analysis procedures: The scope and depth of the review or analysis methods to be used, in addition to the description in the Tool Verification Plan.

b. Test cases: The purpose of each test case, set of inputs, conditions, and expected results to achieve the pass/fail criteria.

c. Test procedures: The step-by-step instructions for how each test case is to be set up and executed, selected inputs, how the test results are evaluated, and the test environment to be used.

10.3.4 Tool Operational Verification and Validation Results

The tool operational verification and validation process produces the Tool Operational Verification and Validation Results, which should:

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

a. For each review, analysis, and test, indicate each procedure that passed or failed during the activities and the final pass/fail results.

b. Identify the configuration item or tool version reviewed, analyzed, or tested.

c. Include the results of tests, reviews, and analyses.

d. Record and track any discrepancies found using the problem reporting process.

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

11.0 ADDITIONAL CONSIDERATIONS FOR TOOL QUALIFICATION

This section provides guidance on additional considerations concerning tool qualification in relation to the use of multi-function tools, previously qualified tools, COTS tools, service history, and alternative methods for achieving compliance with objectives documented in previous sections of this document.

11.1 Multi-Function Tools

A multi-function tool is one that performs multiple functions. Examples of multi-function tool types include:

- A tool with a single executable file with multiple functions.
- A tool composed of multiple executable files that allows disabling of certain executable files.
- A tool packaged in some other way that allows a user to disable or select part of the tool functionality.

Some of the functions may be used to eliminate, reduce, or automate the software life cycle process(es) and some functions may not even be used in a given project. For example, a tool may include multiple development and verification functions; some of the development and verification functions may be used, while other functions are not needed for the specific project.

The appropriate TQL should be determined for each function of the multi-function tool, using the domain-related guidance. The additional guidance below should be applied when qualifying only some functions, or when different TQLs are applied on some functions. Otherwise, the entire tool should be qualified to the highest TQL.

a. Some functions may be qualified at a lower TQL. This is only feasible if the function(s) at the higher level is protected from the function (or group of functions) at the lower TQL(s). The applicant should assess the overall tool functionality and implementation and identify each function (or group of functions) with the proposed TQL(s). This assessment includes an identification of each tool function, the relationship and interaction of tool functions, the tool's operational usage, tool configuration(s), and identification of functions intended to be qualified.

b. Protection between the functions (or groups of functions) with different TQLs should be demonstrated. Acceptable evidence of this protection would be to show the outputs of the functions (or groups of functions) qualified at a lower TQL have no effect on the output of the other functions (that is, the tool capabilities are functionally isolated). The protection approach should be substantiated for each tool function when different TQLs are implemented.

c. When protection between functions (or groups of functions) of different TQLs is shown, the protected functions may be qualified as if they were separate tools.

d. If a multi-function tool both produces an output and verifies this same output, it should be demonstrated that the functions of the tool have been developed using protection techniques to avoid an error that might affect both functions. If the verification objective satisfied by the tool requires independence, the function that

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

verifies the output should be shown to be independent of the function that produces the output. Independence is typically used to ensure that common errors do not exist between the tool functions and to verify that the verification function completely detects any errors in the output.

e. When applying this guidance, some functions of a tool may not be qualified. This is typically the case when some function(s) in a tool are not used in the software life cycle process or the tool output is verified as requested by the domain-related guidance. The qualified function should be shown not to depend on the unused function(s) (or group(s) of functions). Additionally, the unused functions should be clearly identified to ensure they are not inadvertently assumed to be qualified in future qualification efforts.

All multi-function tools should be described in the appropriate tool plans. The plans should explain the tool functionality, the assessment of the TQLs, a description of functions that will and will not be used, and an explanation of how the above guidance will be satisfied.

Note: If the tool does not contain functionality above TQL-5, only section 11.1.d applies.

11.2 Previously Qualified Tools

Previously qualified tools encompass any tool already qualified and intended to be used on a subsequent project (a subsequent project may be a new project or a modification to an existing project). The sections below provide guidance for three aspects of tool qualification: (1) reuse of previously qualified tools that are unchanged, (2) changes to the tool operational environment, and (3) changes to the tool itself. Figure 11-1 depicts the three aspects of previously qualified tools and the order of evaluation.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. Not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

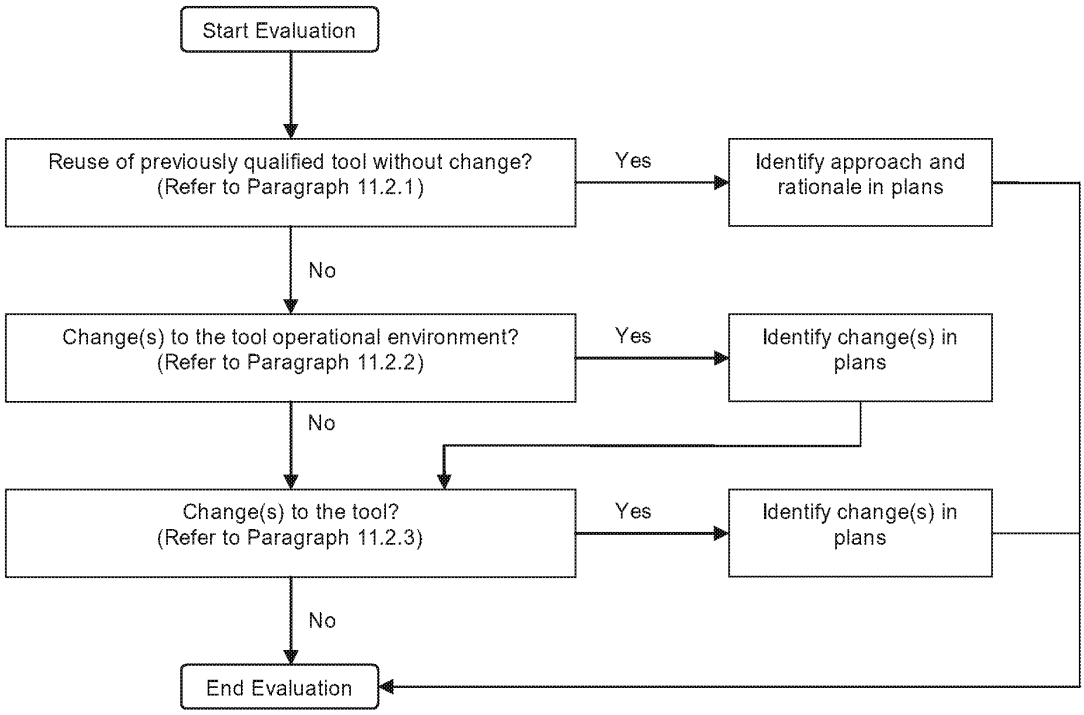


Figure 11-1 Previously Qualified Tools Evaluation

11.2.1 Reuse of Previously Qualified Tools

This section addresses the reuse of qualification data for a tool qualified on a previous project that will be used on a subsequent project without any change.

A tool is suitable for reuse if the following criteria are satisfied:

a. The tool was previously qualified in the scope of previous software projects already approved by the certification authority.

b. The TQL for the reuse is the same or lower than the previous qualification.

c. The tool life cycle data being considered for reuse has not changed since the tool's previous qualification.

d. The tool operational environment is shown to be equivalent to that of the previous qualification.

e. The Tool Operational Requirements are the same as for the previous qualification.

f. The applicant has access to the approved tool qualification data.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

g. The applicant ensures that the same version of the tool is being used as that supported by the qualification data.

If any of the criteria are not satisfied, then sections 11.2.2 and/or 11.2.3 should be analyzed for the change(s).

If all the criteria are satisfied, the intention and rationale for reuse of the tool data should be documented in the PSAC and/or TQP and submitted to the certification authority. In addition, any Tool Problem Report still outstanding or raised (even if subsequently closed) since the previous qualification should be assessed and properly dispositioned.

11.2.2 Changes to the Tool Operational Environment

This section addresses change(s) in the tool operational environment. An example is an upgrade of the operating system of the workstation where the tool is used.

An analysis of the changes should be performed to ensure that:

a. The tool verification environment(s) is representative of the tool operational environment (see 4.4.c).

b. The tool operational environment compatibility requirements described in the Tool Operational Requirements are complete and applicable to the new tool operational environment (see 6.2.2.a.2).

c. The tool installed in the new tool operational environment complies with the need of the software life cycle processes (see 6.2.2.c).

The analysis identifies the activities that need to be performed or re-performed.

A description of the change(s) since the previous qualification and the supporting analysis should be stated in the PSAC and/or TQP and submitted to the certification authority.

11.2.3 Changes to Previously Qualified Tool

This section addresses the changes to a previously qualified tool on a subsequent project.

If there are changes to the tool, tool plans, required TQL, or the intended use of the tool, a change impact analysis is required. The analysis should consider the impact of the change to the following areas, as a minimum:

a. Tool Operational Requirements.

b. Tool Requirements.

c. Tool Design Description.

d. Tool Source Code.

e. Tool development environment and development process.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. Not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

This impact analysis should identify any needed re-verification activities. For example, traceability analysis, data coupling and control coupling analysis, regression testing, requirements review, etc.

The intention, substantiation, and description of the change(s) should be stated in the plans and submitted to the certification authority.

11.3 Qualifying COTS Tools

This section adapts the tool qualification guidance of this document for application to a COTS tool. A COTS tool is typically developed independent of any specific software project, and is intended to be utilized by multiple users.

COTS tools designed to be configurable by the tool user as described in the user's manual are covered under the guidance of this section. Tools that are modified by the tool user are not.

COTS tools, as any other tool to be qualified, need to comply with the objectives appropriate to the necessary TQL. However, to satisfy the objective of the different processes (the tool operational development process, the tool development process, and all the related verification processes), a separation of responsibility between the tool user and the tool developer should be proposed.

11.3.1 Approach for COTS Tool Qualification

COTS tools are typically developed without available Tool Operational Requirements specific to software life cycle. In order to communicate the functionality and intended use of the tool to the users, the tool developer should provide a developer-TOR which includes a set of operational requirements for possible software life cycles. The tool user will use the developer-TOR to determine if the tool meets the software life cycle needs. The tool user provides the TOR identifying the requirements for the operational use of the tool. The TOR is typically above and beyond the developer-TOR.

For COTS tools, the tool qualification activities are typically performed by two entities:

- Tool developer: The entity that developed the tool.
- Tool user: The entity that uses the tool in the scope of a given software project.

Activities are defined below for both the COTS tool developer and tool user in order to satisfy all of the tool qualification objectives. It should be noted that for TQL-5 qualification, objectives may be satisfied without any tool qualification data from the tool developer.

11.3.2 Activities for the Tool Developer

In order to provide a qualifiable tool, the tool developer should:

- Use the guidance of this document to perform the activities and produce the tool life cycle data necessary to demonstrate that all objectives allocated to the tool developer have been satisfied.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

- Provide a developer-TOR to identify the functions implemented in the tool (see 11.3.2.1).
- Provide a set of preliminary tool qualification data appropriate to the TQL (see 11.3.2.2 through 11.3.2.4).

Table 11-1 identifies the objectives typically applicable to the tool developer.

Table 11-1 Typical Tool Developer Objetives

Annex A Table	Objectives Applicable to Tool Developer	Notes
T-0	Objectives 2, 4 and 5: Applicable Others: Not applicable	• Objective 2: Tool developer provides a developer-TOR. • Objectives 4 and 5 are performed using the developer-TOR. • Other objectives: These are the tool user's responsibility.
T-1	All: Applicable	• Objectives 3 and 5: Fully applicable for the tool developer. • Other objectives: The scope of these objectives is limited to the specific activities performed by the tool developer.
T-2	All: Applicable	• The tool developer uses the developer-TOR to perform the activities of the tool development process.
T-3	All: Applicable	• Objectives 1 and 8 are performed using the developer-TOR.
T-4 to T-7	All: Applicable	• The tool developer performs all activities.
T-8	All: Applicable	• Scope is limited to the data produced during the activities performed by the tool developer.
T-9	All: Applicable	• Scope is limited to the activities performed by the tool developer.
T-10	All: Not applicable	• Tool user performs these activities.

11.3.2.1 Developer-TOR

The developer-TOR should be developed so that the tool user can determine if the tool meets the user's needs. The developer-TOR should contain data that the tool user may directly reference from their TOR without modification.

The developer-TOR is used by the tool developer in lieu of the TOR described in this document.

The developer-TOR should include:

a. Description of all anticipated tool operational environments.

b. Description of input files, including format, language definition, etc. (see 10.3.1.c).

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. Not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

c. Description of output files, including format and contents (see 10.3.1.d).

d. Requirements for all expected functional behavior of the tool (from software life cycle process point of view) (see 10.3.1.e).

e. Requirements to address the abnormal activation modes or inconsistency inputs that should be detected by the tool. These requirements should consider the impact of those modes on the functionality and outputs of the tools (see 10.3.1.f).

f. The applicable user information, such as a user manual or installation guide (or a reference to it, if not provided as part of the Tool Requirements data) (see 10.3.1.g).

11.3.2.2 Developer-TQP

The developer-TQP should have the same content as specified in section 10.1.2, but should be limited to those activities allocated to the tool developer. Additionally, the developer-TQP should describe how the preliminary qualification data may be used by a tool user.

11.3.2.3 Developer-TCI

The developer-TCI should have the same content as specified in section 10.1.11, but should be limited to the data provided during activities performed by the tool developer.

11.3.2.4 Developer-TAS

The developer-TAS should have the same content as specified in section 10.1.15, but should be limited to those activities allocated to the tool developer.

11.3.3 Tool Qualification Activities for the Tool User

The selection of COTS tool is typically performed during the software planning process. During this process the user should assess the COTS tool for its ability to meet the needs of the proposed software life cycle process and the compatibility with its tool operational environment. Availability and relevance of COTS tool life cycle data and preliminary set of qualification data to support the appropriate TQL qualification should also be assessed in the context of the software project.

A COTS tool that has been previously qualified is not automatically considered to be qualified for other projects requiring approval (see 11.2 for previously qualified tools guidance).

If existing tool life cycle data is insufficient to demonstrate the compliance to all the objectives applicable to the tool developer, then the tool user should augment the data and/or propose alternate means (see 11.5). This approach should be described in the TQP.

In order to assess the COTS tool in the tool operational environment, the tool user should supplement the developer activities. Table 11-2 identifies the objectives typically applicable to the tool user.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

Table 11-2 Typical Tool User Objectives

Annex A Table	Objectives Applicable to Tool User	Notes
T-0	Objectives 4 and 5: Not applicable Others: Applicable	• Objective 2: Tool user should provide a TOR that supplements the developer-TOR provided by the tool developer. • Objectives 4 and 5: Tool user should evaluate the developer-TOR for input to their TOR. • Objective 6: The tool user should verify the TOR to assess that it satisfies the user's needs. • Objective 7: This activity is performed using the TOR.
T-1	Objectives 3 and 5: Not applicable Others: Applicable	• Objectives 3 and 5: Tool developer activities. • Other objectives: The tool user supplements the data produced by the tool developer to address the tool user's activities. Coordination between tool developer and tool user activities should also be planned.
T-2 to T-7	All: Not Applicable	• All of these are the tool developer's activities.
T-8	All: Applicable	• Scope applies to the data produced during the activities performed by the tool user.
T-9	All: Applicable	• Objectives 1 and 2: Scope is limited to the activities performed by the tool user. • Objective 3: The tool conformity is performed in the scope of a project by the applicant or tool user.
T-10	All: Applicable	• The tool user performs these activities.

For TOR, TQP, TCI, and TAS, the tool user should consider the recommendations in sections 11.3.3.1 through 11.3.3.4.

11.3.3.1 Tool Operational Requirements (TOR)

The tool user should use the developer-TOR to determine if the tool meets the software life cycle needs. If suitable, the tool user should reference or incorporate the developer-TOR data in the TOR and should provide specific information, limitations, etc. for the software life cycle.

The purpose of the TOR is to identify the requirements for the tool use in the software life cycle. The TOR is used as the basis for the compliance verification activities of the tool in the tool operational environment.

The TOR should include but not be limited to the following:

- Developer-TOR contents or reference to applicable developer-TOR contents.
- Description of the context of the tool usage, including interfaces with other tools and integration of the tool output files in the resultant software (see 10.3.1.a).

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. Not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

- Description of the tool operational environment (where the tool will be installed). This includes an analysis of the anticipated tool operational environment(s) included in the developer-TOR (see 10.3.1.b).

- Description of the operational usage of the tool (including, selected options, parameters values, command line) (see 10.3.1.h).

- Description of the operational inputs of the tool in the specific operational environment. The purpose of this description is to identify all possible input data and its associated structure. The completeness and accuracy of this description will allow development of the tool operational verification data that encompasses the operational usage of the tool (see 10.3.1.c).

- Performance requirements specifying the behavior of tool outputs (see 10.3.1.i).

11.3.3.2 Tool Qualification Plan (TQP)

In addition to the information provided in section 10.1.2, the TQP should provide:

- The reference to the developer-TQP.

- The division of responsibilities between tool developer and tool user for the objectives and activities.

11.3.3.3 Tool Configuration Index (TCI)

In addition to the information provided in section 10.1.11, the TCI should provide:

- The reference to the developer-TCI.

- The identification of the data produced by the tool user.

11.3.3.4 Tool Accomplishment Summary (TAS)

In addition to the information provided in section 10.1.15, the TAS should provide:

- The reference to the developer-TAS.

- The identification of all options that have to be used during the execution of the tool.

- The demonstration (analysis and/or review) that the tool installation operation and the tool configuration operation performed by the tool user are in accordance with the recommendations of the user's manual provided by the tool developer.

- Deviations from TQP in the division of responsibilities between tool developer and tool user for the objectives and activities.

- The results for both the tool developer's and the tool user's verification activities.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

The tool user should make all qualification data available to the certification authority. Typically, the tool user will integrate or reference the preliminary data provided by the tool developer in their final tool qualification life cycle data.

11.3.4 Coordination of Developer and User Activities

Objectives that are performed by the tool developer will be assessed by the tool user, since the tool user is ultimately responsible for qualification.

The TQP should reference the developer-TQP and provide the means for coordination. The same is true for the TAS and TCI referencing the developer-TAS and developer-TCI.

During the tool conformity review the tool user should analyze the tool developer's conformity review report and verify, if necessary, the status of corrective actions raised during this review.

11.4 Service History

Once the applicant has identified or developed the Tool Operational Requirements, tool service history may be used to satisfy objectives of this document. If service history is used, its relevance should be substantiated. Service history is assessed on a case-by-case basis.

Some possible reasons to use service history include:

- The tool has been used but all the data needed for qualification is not available.
- The TQL needs to be increased.
- Some aspects of the tool are difficult to analyze using the existing data.
- The confidence in the compliance of the tool to Tool Operational Requirements needs to be increased.

11.4.1 Aspects of Tool Service History

The following aspects of tool service history should be evaluated:

a. Available data: Identify the available tool service history data and its relevance to demonstrate compliance to the Tool Operational Requirements.

b. Problem reporting: Problem reporting and TCM are required, as a minimum, to support service history. The problem reporting should include consistent definition of the problems, classification of problems, tracking with respect to tool versions, tracking with respect to solutions, and identification of the scope of the problem reporting system (for example, the number of participating applicants).

c. Environment: The similarity of the tool operational environment in which the tool service history data was collected to the one used in the proposed project is critical for establishing the tool service history credit. Environment analysis should include any restrictions on tool use imposed by the tool's limitations.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. Not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

d. Operation: Operating modes or settings, interactions with users, and procedures are fundamental to show relevance of tool service history for the proposed tool application. The level of experience and/or training in the use of the tool on the part of the applicant should also be considered. Operation analysis should include any restrictions on tool use imposed by the applicant.

e. Stability and maturity: The tool's change history should be assessed to determine in-service stability and maturity. An analysis should be performed to assess the impact of the change on the tool or the tool operational environment to determine if the service history duration should be reset.

f. Application service: The service history of the applications developed using the tool can provide additional substantiation of the tool's stability and maturity.

11.4.2 Tool Service History Activities

Activities for the use of tool service history include:

a. The applicant should show that the tool service history data demonstrates compliance to the Tool Operational Requirements.

b. The applicant should show that the tool operational environment has been identified, documented, and maintained throughout the period of use for which certification credit will be sought.

c. The applicant should show that the problem reporting is established during the tool's service history in order to ensure that:

1. Representative data is available.

2. Tool problems were reported and the resulting actions recorded.

3. Tool problems were analyzed to determine how they occurred and which problems were corrected. The analysis should show if the reported tool problem has a possible impact to the tool output and if an acceptable work-around exists.

d. Configuration changes to the tool during the tool's service history should be identified and the effect analyzed to confirm the stability and maturity of the tool.

e. An analysis of tool operational environment should be performed to ensure tool service history can be applied.

f. Service history of a tool may be used to support the qualification of a subset of a tool. The guidance for qualification of multi-function tools should be considered (see 11.1).

g. The PSAC or TQP should describe the intended tool use and should include the following:

1. Summary of service history data.

2. Rationale for use of tool's service history and how it relates to the TQL.

3. Analysis of the relevance of the tool's service history environment.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

4. Availability of problem reporting during the tool service history.

5. The tool change history to evaluate the stability and maturity of the tool.

11.5 Alternative Methods for Tool Qualification

Applicants may decide to use an alternative method for tool qualification. Potential alternative methods include but are not limited to exhaustive input testing, formal methods, and dissimilar tools (for example, two or more tools developed separately that satisfy the same operational requirements).

An alternative method should be considered in the context of the overall software life cycle process. The effort for obtaining certification credit of an alternative method is dependent on the software level and the impact of the alternative method on the software life cycle processes. Activities for using an alternative method include:

a. The tool plans should describe the application of the alternative method to the tool life cycle processes and outputs. The alternative method should then provide evidence of correctness of implementation of the method.

b. The applicant should specify in the PSAC or TQP, and obtain agreement from the certification authority for:

   1. The impact of the proposed alternative method on the software life cycle processes.

   2. The impact of the proposed alternative method on the software life cycle data.

   3. The rationale (including description, justification, and substantiation) for use of the alternative method which shows how the objectives of this document are satisfied.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. Not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

ANNEX A TOOL QUALIFICATION OBJECTIVES

These tables summarize the objectives of this document by showing the applicability of each objective by TQL, whether the objective is to be implemented with independence, the output which results from satisfying the objective, and the control category for each tool life cycle data item. It should be noted that the activities which are identified to satisfy the objectives are also included in the tables for reference (in the "Activity" column).

These tables should not be used as a checklist. These tables do not reflect all aspects of compliance to this document. In order to fully understand the guidance, the full body of this document should be considered.

An overview of each table is provided below:

- Table T-0 addresses the tool operational processes, including the Tool Operational Requirements process, tool operational integration process, and the tool operational verification and validation process.
- Table T-1 addresses the tool planning process of the tool.
- Table T-2 addresses the objectives applicable to the tool development processes.
- Tables T-3 to T-7 address the tool verification processes.
- Table T-8 addresses the tool configuration management process and is applicable to the complete tool life cycle.
- Table T-9 addresses the tool quality assurance process and is applicable to the complete tool life cycle.
- Table T-10 adresses the tool qualification liaison process.

The following legend applies to "Applicability by TQL" and "Control Category by TQL" for all tables:

LEGEND:		
	●	The objective should be satisfied with independence.
	○	The objective should be satisfied.
	Blank	Satisfaction of objective is at applicant's discretion.
	①	Data satisfies the objectives of Control Category 1 (CC1).
	②	Data satisfies the objectives of Control Category 2 (CC2).

For Personal Use by Jordan Colson, Skyryse © 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Table T-0 Tool Operational Processes

	Objective		Activity	Applicability by TQL					Output		Control Category by TQL				
	Description	Ref.	Ref.	1	2	3	4	5	Description	Ref.	1	2	3	4	5
	Planning Process														
1	The tool qualification need is established.	4.1	[Note 1]	❍	❍	❍	❍	❍	Tool-specific information in the Plan for Software Aspects of Certification	10.1.1	①	①	①	①	①
	Tool Operational Requirements Process														
2	Tool Operational Requirements are defined.	5.1.1.a	5.1.2.a 5.1.2.b 5.1.2.c	❍	❍	❍	❍	❍	Tool Operational Requirements	10.3.1	①	①	①	①	②
	Tool Operational Integration Process														
3	Tool Executable Object Code is installed in the tool operational environment.	5.3.1.a	5.3.2.a 5.3.2.b 5.3.2.c	❍	❍	❍	❍	❍	Tool Executable Object Code Tool Installation Report	10.2.4 10.3.2	② ②	② ②	② ②	② ②	② ②
	Tool Operational Verification and Validation Process														
4	Tool Operational Requirements are complete, accurate, verifiable, and consistent.	6.2.1.a	6.2.2.a	●	●	❍	❍		Tool Operational Verification and Validation Results	10.3.4	②	②	②	②	
5	Tool operation complies with the Tool Operational Requirements.	6.2.1.b	6.2.2.c	●	●	❍	❍	❍	Tool Operational Verification and Validation Cases and Procedures Tool Operational Verification and Validation Results	10.3.3 10.3.4	② ②	② ②	② ②	② ②	② ②
6	Tool Operational Requirements are sufficient and correct.	6.2.1.aa	6.2.2.b	●	●	❍	❍	❍	Tool Operational Verification and Validation Results	10.3.4	②	②	②	②	②
7	Software life cycle process needs are met by the tool.	6.2.1.bb	6.2.2.c	❍	❍	❍	❍	❍	Tool Operational Verification and Validation Cases and Procedures Tool Operational Verification and Validation Results	10.3.3 10.3.4	② ②	② ②	② ②	② ②	② ②

Note:
1. This activity is part of the software planning process and is not fully described in this document. Refer to the domain document (for example, DO-178C or DO-278A).

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Table T-1 Tool Planning Process

	Objective		Activity	Applicability by TQL					Output		Control Category by TQL				
	Description	Ref.	Ref.	1	2	3	4	5	Description	Ref.	1	2	3	4	5
1	Tool development and integral processes are defined.	4.3.a	4.4.a	❍	❍	❍	❍		Tool Qualification Plan	10.1.2	①	①	①	①	
									Tool Development Plan	10.1.3	①	①	②	②	
2	Transition criteria, inter-relationships, and sequencing among processes of tool processes are defined.	4.3.b	4.4.a	❍	❍	❍			Tool Verification Plan	10.1.4	①	①	②	②	
									Tool Configuration Management Plan	10.1.5	①	①	②	②	
3	Tool development environment is selected and defined.	4.3.c	4.4.c	❍	❍	❍									
									Tool Quality Assurance Plan	10.1.6	①	①	②	②	
4	Additional considerations are addressed.	4.3.d	4.4.a 4.4.e	❍	❍	❍	❍								
5	Tool development standards are defined.	4.3.e	4.4.b	❍	❍	❍			Tool Requirements Standards	10.1.7	①	①	②		
									Tool Design Standards	10.1.8	①	①	②		
									Tool Code Standards	10.1.9	①	①	②		
6	Tool plans comply with this document.	4.3.f	4.4.d	❍	❍	❍			Tool Verification Results	10.2.6	②	②	②		
7	Development and revision of tool plans are coordinated.	4.3.g	4.4.d	❍	❍	❍			Tool Verification Results	10.2.6	②	②	②		

For Personal Use by Jordan Colson, Skyryse © 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Table T-2 Tool Development Processes

	Objective		Activity	Applicability by TQL					Output		Control Category by TQL				
	Description	Ref.	Ref.	1	2	3	4	5	Description	Ref.	1	2	3	4	5
1	Tool Requirements are developed.	5.2.1.1.a	5.2.1.2.a 5.2.1.2.b 5.2.1.2.c 5.2.1.2.d 5.2.1.2.e 5.2.1.2.f 5.2.1.2.g 5.2.1.2.h 5.2.1.2.i 5.2.1.2.j 5.2.1.2.k 5.2.5.a	❍	❍	❍	❍		Tool Requirements Trace Data	10.2.1 10.2.7	① ①	① ①	① ①	① ①	
2	Derived tool requirements are defined.	5.2.1.1.b	5.2.1.2.h	❍	❍	❍	❍		Tool Requirements	10.2.1	①	①	①	①	
3	Tool architecture is developed.	5.2.2.1.a	5.2.2.2.a 5.2.2.2.b 5.2.2.2.e 5.2.2.2.f 5.2.2.2.g	❍	❍	❍	❍		Tool Design Description	10.2.2	①	①	①	②	
4	Low-level tool requirements are developed.	5.2.2.1.b	5.2.2.2.c 5.2.2.2.e 5.2.2.2.f 5.2.5.b	❍	❍	❍			Tool Design Description Trace Data	10.2.2 10.2.7	① ①	① ①	① ①		
5	Derived low-level tool requirements are defined.	5.2.2.1.c	5.2.2.2.c 5.2.2.2.d	❍	❍	❍			Tool Design Description	10.2.2	①	①	①		
6	Tool Source Code is developed.	5.2.3.1.a	5.2.3.2.a 5.2.3.2.b 5.3.3.2.c 5.2.3.2.d 5.2.5.c	❍	❍	❍			Tool Source Code Trace Data	10.2.3 10.2.7	① ①	① ①	① ①		
7	Tool Executable Object Code is produced.	5.2.4.1.a	5.2.4.2.a 5.2.4.2.b	❍	❍	❍	❍		Tool Executable Object Code	10.2.4	①	①	①	①	
8	Tool is installed in the tool verification environment(s).	5.2.4.1.b	5.2.4.2.c	❍	❍	❍	❍		Tool Executable Object Code	10.2.4	①	①	①	①	

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Table T-3 Verification of Outputs of Tool Requirements Processes

	Objective		Activity	Applicability by TQL					Output		Control Category by TQL				
	Description	Ref.	Ref.	1	2	3	4	5	Description	Ref.	1	2	3	4	5
1	Tool Requirements comply with Tool Operational Requirements.	6.1.3.1.a	6.1.3.1	●	●	○	○		Tool Verification Results	10.2.6	②	②	②	②	
2	Tool Requirements are accurate and consistent.	6.1.3.1.b	6.1.3.1	●	●	○	○		Tool Verification Results	10.2.6	②	②	②	②	
3	Requirements for compatibility with the tool operational environment are defined.	6.1.3.1.c	6.1.3.1	●	●	○	○		Tool Verification Results	10.2.6	②	②	②	②	
4	Tool Requirements define the behavior of the tool in response to error conditions.	6.1.3.1.d	6.1.3.1	●	●	○	○		Tool Verification Results	10.2.6	②	②	②	②	
5	Tool Requirements define user instructions and error messages.	6.1.3.1.e	6.1.3.1	●	●	○	○		Tool Verification Results	10.2.6	②	②	②	②	
6	Tool Requirements are verifiable.	6.1.3.1.f	6.1.3.1	○	○	○	○		Tool Verification Results	10.2.6	②	②	②	②	
7	Tool Requirements conform to Tool Requirements Standards.	6.1.3.1.g	6.1.3.1	○	○	○			Tool Verification Results	10.2.6	②	②	②		
8	Tool Requirements are traceable to Tool Operational Requirements.	6.1.3.1.h	6.1.3.1	○	○	○	○		Tool Verification Results	10.2.6	②	②	②	②	
9	Algorithms are accurate.	6.1.3.1.i	6.1.3.1	●	●	○	○		Tool Verification Results	10.2.6	②	②	②	②	

For Personal Use by Jordan Colson, Skyryse © 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Table T-4 Verification of Outputs of Tool Design Process

	Objective		Activity	Applicability by TQL					Output		Control Category by TQL				
	Description	Ref.	Ref.	1	2	3	4	5	Description	Ref.	1	2	3	4	5
1	Low-Level Tool requirements comply with Tool Requirements.	6.1.3.2.a	6.1.3.2	●	●	❍			Tool Verification Results	10.2.6	②	②	②		
2	Low-level tool requirements are accurate and consistent.	6.1.3.2.b	6.1.3.2	●	●	❍			Tool Verification Results	10.2.6	②	②	②		
3	Low-level tool requirements are verifiable.	6.1.3.2.c	6.1.3.2	❍	❍				Tool Verification Results	10.2.6	②	②			
4	Low-level tool requirements conform to Tool Design Standards.	6.1.3.2.d	6.1.3.2	❍	❍	❍			Tool Verification Results	10.2.6	②	②	②		
5	Low-level tool requirements are traceable to Tool Requirements.	6.1.3.2.e	6.1.3.2	❍	❍	❍			Tool Verification Results	10.2.6	②	②	②		
6	Algorithms are accurate.	6.1.3.2.f	6.1.3.2	●	●	❍			Tool Verification Results	10.2.6	②	②	②		
7	Tool architecture is compatible with Tool Requirements.	6.1.3.3.a	6.1.3.3	●	❍	❍			Tool Verification Results	10.2.6	②	②	②		
8	Tool architecture is consistent.	6.1.3.3.b	6.1.3.3	●	❍	❍			Tool Verification Results	10.2.6	②	②	②		
9	Tool architecture conforms to Tool Design Standards.	6.1.3.3.c	6.1.3.3	❍	❍	❍			Tool Verification Results	10.2.6	②	②	②		
10	Protection mechanisms, if used, are confirmed.	6.1.3.3.d	6.1.3.3	●	❍	❍	❍		Tool Verification Results	10.2.6	②	②	②	②	
11	External component interface is correctly and completely defined.	6.1.3.3.e	6.1.3.3	●	❍				Tool Verification Results	10.2.6	②	②			

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Table T-5 Verification of Outputs of Tool Coding & Integration Process

	Objective		Activity	Applicability by TQL					Output		Control Category by TQL				
	Description	Ref.		1	2	3	4	5	Description	Ref.	1	2	3	4	5
1	Tool Source Code complies with low-level tool requirements.	6.1.3.4.a	6.1.3.4	●	●	❍			Tool Verification Results	10.2.6	②	②	②		
2	Tool Source Code complies with tool architecture.	6.1.3.4.b	6.1.3.4	●	❍	❍			Tool Verification Results	10.2.6	②	②	②		
3	Tool Source Code is verifiable.	6.1.3.4.c	6.1.3.4	❍	❍				Tool Verification Results	10.2.6	②	②			
4	Tool Source Code conforms to Tool Code Standards.	6.1.3.4.d	6.1.3.4	❍	❍	❍			Tool Verification Results	10.2.6	②	②	②		
5	Tool Source Code is traceable to low-level tool requirements.	6.1.3.4.e	6.1.3.4	❍	❍	❍			Tool Verification Results	10.2.6	②	②	②		
6	Source Code is accurate and consistent.	6.1.3.4.f	6.1.3.4	●	❍	❍			Tool Verification Results	10.2.6	②	②	②		
7	Output of tool integration process is complete and correct.	6.1.3.5.a	6.1.3.5	❍	❍	❍			Tool Verification Results	10.2.6	②	②	②		

For Personal Use by Jordan Colson, Skyryse © 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Table T-6 Testing of Outputs of Integration Process

	Objective		Activity	Applicability by TQL					Output		Control Category by TQL				
	Description	Ref.	Ref.	1	2	3	4	5	Description	Ref.	1	2	3	4	5
1	Tool Executable Object Code complies with Tool Requirements.	6.1.4.1.a	6.1.4.2.a 6.1.4.2.b 6.1.5	❍	❍	❍	❍		Tool Verification Cases and Procedures Tool Verification Results Trace Data	10.2.5 10.2.6 10.2.7	① ② ①	① ② ①	② ② ②	② ② ②	
2	Tool Executable Object Code is robust with Tool Requirements.	6.1.4.1.b	6.1.4.2.a 6.1.4.2.c 6.1.4.2.d 6.1.5	❍	❍	❍	❍		Tool Verification Cases and Procedures Tool Verification Results Trace Data	10.2.5 10.2.6 10.2.7	① ② ①	① ② ①	② ② ②	② ② ②	
3	Tool Executable Object Code complies with low-level tool requirements.	6.1.4.1.c	6.1.4.2.a 6.1.4.2.b 6.1.5	●	●	❍			Tool Verification Cases and Procedures Tool Verification Results Trace Data	10.2.5 10.2.6 10.2.7	① ② ①	① ② ①	② ② ②		
4	Tool Executable Object Code is robust with low-level tool requirements.	6.1.4.1.d	6.1.4.2.a 6.1.4.2.c 6.1.4.2.d 6.1.5	●	❍	❍			Tool Verification Cases and Procedures Tool Verification Results Trace Data	10.2.5 10.2.6 10.2.7	① ② ①	① ② ①	② ② ②		

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Table T-7 Verification of Outputs of Tool Testing

	Objective		Activity	Applicability by TQL					Output		Control Category by TQL				
	Description	Ref.	Ref.	1	2	3	4	5	Description	Ref.	1	2	3	4	5
1	Test procedures are correct.	6.1.4.4.b	6.1.4.4	●	❍	❍			Tool Verification Results	10.2.6	②	②	②		
2	Test results are correct and discrepancies explained.	6.1.4.4.c	6.1.4.4	●	❍	❍	❍		Tool Verification Results	10.2.6	②	②	②	②	
3	Test coverage of Tool Requirements is achieved.	6.1.4.4.a	6.1.4.4	●	❍	❍	❍		Tool Verification Results	10.2.6	②	②	②	②	
4	Test coverage of low-level tool requirements is achieved.	6.1.4.4.a	6.1.4.4	●	❍	❍			Tool Verification Results	10.2.6	②	②	②		
5	Analysis of requirements-based testing of external components is achieved.	6.1.4.3.1	6.1.4.3.2 6.1.4.3.3	●					Tool Verification Results	10.2.6	②				
6	Analysis of requirements-based testing (structural coverage to the level of MC/DC) is achieved.	6.1.4.3.1	6.1.4.3.2 6.1.4.3.3	●					Tool Verification Results	10.2.6	②				
7	Analysis of requirements-based testing (structural coverage to the level of decision coverage) is achieved.	6.1.4.3.1	6.1.4.3.2 6.1.4.3.3	●	●				Tool Verification Results	10.2.6	②	②			
8	Analysis of requirements-based testing (structural coverage to the level of statement coverage) is achieved.	6.1.4.3.1	6.1.4.3.2 6.1.4.3.3	●	●	❍			Tool Verification Results	10.2.6	②	②	②		
9	Analysis of requirements-based testing (data coupling and control coupling) is achieved.	6.1.4.3.1	6.1.4.3.2 6.1.4.3.3	●	●	❍			Tool Verification Results	10.2.6	②	②	②		

For Personal Use by Jordan Colson, Skyryse © 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Table T-8 Tool Configuration Management Process

	Objective		Activity	Applicability by TQL					Output		Control Category by TQL				
	Description	Ref.	Ref.	1	2	3	4	5	Description	Ref.	1	2	3	4	5
1	Configuration items are identified.	7.1.a	7.2.1	❍	❍	❍	❍	❍	Tool Configuration Management Records	10.1.13	②	②	②	②	②
2	Baselines and traceability are established.	7.1.b	7.2.2	❍	❍	❍	❍		Tool Configuration Index Tool Configuration Management Records	10.1.11 10.1.13	① ②	① ②	① ②	① ②	
3	Problem reporting, change control, change review, and configuration status accounting are established.	7.1.c 7.1.d 7.1.e 7.1.f	7.2.3 7.2.4 7.2.5 7.2.6	❍	❍	❍	❍		Tool Problem Reports Tool Configuration Management Records	10.1.12 10.1.13	② ②	② ②	② ②	② ②	
4	Archive, retrieval, and release are established.	7.1.g	7.2.7	❍	❍	❍	❍	❍	Tool Configuration Management Records	10.1.13	②	②	②	②	②
5	Tool life cycle environment control is established.	7.1.h	7.4	❍	❍	❍	❍		Tool Configuration Management Records Tool Life Cycle Environment Configuration Index	10.1.13 10.1.10	② ①	② ①	② ①	② ②	

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Table T-9 Tool Quality Assurance Process

	Objective		Activity	Applicability by TQL					Output		Control Category by TQL				
	Description	Ref.	Ref.	1	2	3	4	5	Description	Ref.	1	2	3	4	5
1	Assurance is obtained that tool plans and standards are developed and reviewed for consistency.	8.1.a	8.2.b 8.2.h	●	●	●			Tool Quality Assurance Records	10.1.14	②	②	②		
2	Assurance is obtained that tool processes comply with approved plans.	8.1.b	8.2.a 8.2.c 8.2.d 8.2.f 8.2.h 8.2.i	●	●	●	●	●	Tool Quality Assurance Records [Notes 1 & 2]	10.1.14	②	②	②	②	②
3	Assurance is obtained that tool processes comply with approved standards.	8.1.b	8.2.a 8.2.c 8.2.d 8.2.f 8.2.h 8.2.i	●	●	●			Tool Quality Assurance Records [Note 2]	10.1.14	②	②	②		
4	Assurance is obtained that transition criteria for the tool life cycle processes are satisfied.	8.1.c	8.2.e 8.2.h	●	●	●			Tool Quality Assurance Records	10.1.14	②	②	②		
5	Tool conformity review is conducted.	8.1.d	8.2.g 8.2.h 8.3	●	●	●	●	●	Tool Quality Assurance Records [Note 1]	10.1.14	②	②	②	②	②

Notes:
1. For TQL-5, Tool Quality Assurance Records may be part of the Software Quality Assurance Records.
2. The nature of the approved tool plans and standards varies by TQL.

For Personal Use by Jordan Colson, Skyryse © 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Table T-10 Tool Qualification Liaison Process

	Objective		Activity	Applicability by TQL					Output		Control Category by TQL				
	Description	Ref.	Ref.	1	2	3	4	5	Description	Ref.	1	2	3	4	5
1	Communication and understanding between the applicant and the certification authority is established.	9.0	9.0.a	❍	❍	❍	❍	❍	Tool-specific information in Plan for Software Aspects of Certification Tool Qualification Plan [Note 1]	10.1.1 10.1.2	① ①	① ①	① ①	① ①	① ①
2	The means of compliance is proposed and agreement is obtained.	9.0	9.0.b	❍	❍	❍	❍	❍	Tool-specific information in Plan for Software Aspects of Certification Tool Qualification Plan [Note 1]	10.1.1 10.1.2	① ①	① ①	① ①	① ①	① ①
3	Compliance substantiation is provided.	9.0	9.0.c 9.0.e	❍	❍	❍	❍	❍	Tool-specific information in Software Accomplishment Summary Tool Accomplishment Summary [Note 2] Tool-specific information in Software Life Cycle Environment Configuration Index Tool Configuration Index [Note 3]	10.1.16 10.1.15 10.1.17 10.1.11	① ① ①	① ① ①	① ① ①	① ① ①	① ① ①
4	Impact of known problems on the Tool Operational Requirements is identified and analyzed.	9.0	9.0.d	❍	❍	❍	❍	❍	Tool-specific information in Software Accomplishment Summary	10.1.16	①	①	①	①	①

Notes:
1. For TQL-5, this may be satisfied by the Plan for Software Aspects of Certification.
2. For TQL-5, this may be satisfied by the Software Accomplishment Summary.
3. For TQL-5, this may be satisfied by the Software Life Cycle Environment Configuration Index.

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

ANNEX B ACRONYMS AND GLOSSARY OF TERMS

Acronym	Meaning
ACG	Autocode Generator
AL	Assurance Level
CC1	Control Category 1
CC2	Control Category 2
CNS/ATM	Communication, Navigation, Surveillance, and Air Traffic Management
COTS	Commercial Off-The-Shelf
DO-178B & DO-178C	RTCA document entitled "Software Considerations in Airborne Systems and Equipment Certification"
DO-278A	RTCA document entitled "Software Integrity Assurance Considerations for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems"
DO-330	RTCA document entitled "Software Tool Qualification Considerations"
EUROCAE	European Organisation for Civil Aviation Equipment
FAQ	Frequently Asked Question
MC/DC	Modified Condition / Decision Coverage
PSAA	Plan for Software Aspects of Approval
PSAC	Plan for Software Aspects of Certification
RTCA	RTCA, Inc. is an association of aeronautical organizations in the United States
SAE	Formerly known as Society of Automotive Engineers
SAS	Software Accomplishment Summary
SC-205	RTCA Special Committee #205
SCI	Software Configuration Index
SECI	Software Life Cycle Environment Configuration Index
TAS	Tool Accomplishment Summary
TCI	Tool Configuration Index
TCM	Tool Configuration Management
TOR	Tool Operational Requirements
TQL	Tool Qualification Level
TQA	Tool Quality Assurance
TQP	Tool Qualification Plan
U.S.	United States
V&V	Validation and Verification
WG-71	EUROCAE Working Group #71

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.
© 2011 RTCA, Inc.

GLOSSARY

These definitions are provided for the terms which are used in this document.

Activity – Tasks and design considerations that provide a means of meeting the objectives.

Algorithm – A finite set of well-defined rules that give a sequence of operations for performing a specific task.

Alternative method – Different approach to satisfy one or more objectives of this document.

Anomalous behavior – Behavior that is inconsistent with specified requirements.

Applicant – A person or organization seeking approval from the certification authority.

Approval – The act or instance of giving formal recognition or official sanction.

Approved source – The location of the tool life cycle data to be retrieved is identified in the Tool Configuration Index. The "approved source" could be a configuration management library, an electronic archive, or an organization other than the developing organization.

Assurance – The planned and systematic actions necessary to provide adequate confidence and evidence that a product or process satisfies given requirements.

Audit – An independent examination of the tool life cycle processes and their outputs to confirm required attributes.

Baseline – The approved, recorded configuration of one or more configuration items, that thereafter serves as the basis for further development, and that is changed only through change control procedures.

Boolean operator – An operator with Boolean operands that yields a Boolean result.

Certification credit – Acceptance by the certification authority that a process, product, or demonstration satisfies a certification requirement.

Change control – (1) The process of recording, evaluating, approving or disapproving, and coordinating changes to configuration items after formal establishment of their configuration identification or to baselines after their establishment. (2) The systematic evaluation, coordination, approval or disapproval, and implementation of approved changes in the configuration of a configuration item after formal establishment of its configuration identification or to baselines after their establishment.

> *Note: This term may be called "configuration control" in other industry standards.*

Code – The implementation of particular data or a particular computer program in a symbolic form, such as source code, object code, or machine code.

Compiler – Program that translates source code statements of a high level language, such as FORTRAN or Pascal, into object code.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Component – A self-contained part, combination of parts, subassemblies, or units that performs a distinct function of a system.

Condition – A Boolean expression containing no Boolean operators except for the unary operator, that is, NOT.

Configuration identification – (1) The process of designating the configuration items and recording their characteristics. (2) The approved documentation that defines a configuration item.

Configuration item – (1) One or more software components treated as a unit for configuration management purposes. (2) Tool life cycle data treated as a unit for configuration management purposes.

Configuration management – (1) The process of: (a) identifying and defining the configuration items of a tool product; (b) controlling the release and change of these items throughout the tool life cycle; (c) recording and reporting the status of configuration items and change requests; and (d) verifying the completeness and correctness of configuration items. (2) A discipline applying technical and administrative direction and surveillance to: (a) identify and record the functional and physical characteristics of a configuration item; (b) control changes to those characteristics; and (c) record and report change control processing and implementation status.

Configuration status accounting – The recording and reporting of the information necessary to manage a configuration effectively, including a listing of the approved configuration identification, the status of proposed changes to the configuration and the implementation status of approved changes.

Control category – Configuration management controls placed on tool life cycle data. The two categories, CC1 and CC2, define the tool configuration management processes and activities applied to control tool life cycle data.

Control coupling – The manner or degree by which one software component influences the execution of another software component.

COTS tool – Commercial tools made available by supplier through public catalog listings. A contract-negotiated tool newly developed for a specific software project is not a COTS tool.

Coverage analysis – The process of determining the degree to which a proposed tool verification process activity satisfies its objective.

Data coupling – The dependence of a software component on data not exclusively under the control of that software component.

Deactivated code – Executable object code (or data) that is traceable to a requirement and, by design, is either: (a) not intended to be executed (code) or used (data), for example, a part of a previously developed software component such as unused legacy code, unused library functions, or future growth code; or (b) is only executed (code) or used (data) in certain configurations, for example, code that is enabled by programmed options. The following examples are often mistakenly categorized as deactivated code but should be identified as required for implementation of the design/requirements: defensive programming structures inserted for robustness, for example, compiler inserted object

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

code for range and array index checks, error or exception handling routines, bounds and reasonableness checking, queuing controls, and time stamps.

Dead code – Executable object code (or data) which exists as a result of a tool development error but cannot be executed (code) or used (data) in any operational configuration. It is not traceable to any tool requirement. The following exceptions are often mistakenly categorized as dead code but are necessary for implementation of the requirements/design: defensive programming structures to improve robustness and deactivated code such as unused library functions. Those unused library functions should be properly deactivated, developed at the same level of rigor as the other tool components, and verified accordingly.

Decision – A Boolean expression composed of conditions and zero or more Boolean operators. If a condition appears more than once in a decision, each occurrence is a distinct condition.

Decision coverage – Every point of entry and exit in the program has been invoked at least once and every decision in the program has taken on all possible outcomes at least once.

Derived requirements – Requirements produced by the tool development processes which specify behavior beyond that specified by the tool operational requirements or the higher level requirements and thus are not directly traceable to tool operational requirements or higher level requirements.

Equivalence class – The partition of the input domain of a program such that a test of a representative value of the class is equivalent to a test of other values of the class.

Error – A mistake in requirements, design, or code.

External components – Components of the tool software that are outside the control of the developer of the tool. Examples include primitive functions provided by the operating system or compiler run-time library, or functions provided by a COTS or open source software library.

Independence – Separation of responsibilities which ensures the accomplishment of objective evaluation. (1) For tool verification process activities, independence is achieved when the verification activity is performed by a person(s) other than the developer of the item being verified, and a tool(s) may be used to achieve equivalence to the human verification activity. (2) For the tool quality assurance process, independence also includes the authority to ensure corrective action.

Integral process – A process which assists the tool development processes and other integral processes and, therefore, remains active throughout the tool life cycle. The integral processes are the tool verification process, the tool quality assurance process, the tool configuration management process, and the tool qualification liaison process.

Low-level tool requirements – Requirements derived from Tool Requirements, derived tool requirements, and design constraints from which Tool Source Code can be directly implemented without further information. These requirements address design implementation details. If Tool Source Code may be directly implemented from Tool Requirements, low-level tool requirements and Tool Requirements are equivalent.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Media – Device or material which acts as a means of transferal or storage of software, for example, programmable read-only memory, magnetic tapes or discs, and paper.

Modified condition / decision coverage – Every point of entry and exit in the program has been invoked at least once, every condition in a decision in the program has taken all possible outcomes at least once, every decision in the program has taken all possible outcomes at least once, and each condition in a decision has been shown to independently affect that decision's outcome. A condition is shown to independently affect a decision's outcome by: (1) varying just that condition while holding fixed all other possible conditions, or (2) varying just that condition while holding fixed all other possible conditions that could affect the outcome.

Monitoring – The act of witnessing or inspecting selected instances of test, inspection, or other activity, or records of those activities, to assure that the activity is under control and that the reported results are representative of the expected results. Monitoring is usually associated with activities done over an extended period of time where 100% witnessing is considered impractical or unnecessary. Monitoring permits authentication that the claimed activity was performed as planned.

Objective – When this document is identified as a means of compliance to the regulations, the objectives are requirements that should be met to demonstrate compliance.

Part number – A set of numbers, letters, or other characters used to identify a configuration item.

Previously developed component (in a tool product) – Component already developed for use. This encompasses a wide range of components, including COTS components and components developed to previous or current guidance.

Process – A collection of activities performed in the tool life cycle to produce a definable output or product.

Protection – The use of a mechanism to ensure that a tool function cannot adversely impact another tool function.

Release – The act of formally making available and authorizing the use of a retrievable configuration item.

Reverification – The evaluation of the outputs of a modification process (for example, correction of errors or the introduction of new or additional functionality) to ensure correctness and consistency with respect to the inputs and standards provided to that process.

Robustness – The extent to which tool can operate correctly despite abnormal inputs and conditions.

Service history data – Tool execution data collected during the service history period.

Software tool – See definition for “Tool”.

Standard – A rule or basis of comparison used to provide both guidance in and assessment of the performance of a given activity or the content of a specified data item.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

Statement coverage – Every statement in the program has been invoked at least once.

Note: Statement is as defined by the programming language.

Structural coverage analysis – An evaluation of the code structure, including interfaces, exercised during requirements-based testing.

Test case – A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

Test procedure – Detailed instructions for the set-up and execution of a given set of test cases, and instructions for the evaluation of results of executing the test cases.

Testing – The process of exercising a tool or a tool component to verify that it satisfies specified requirements and to detect errors.

Tool – A computer program or a functional part thereof, used to help develop, transform, test, analyze, produce, or modify another program, data, or its documentation.

Tool architecture – The structure of the tool selected to implement the tool requirements.

Tool conformity review – A review, typically conducted at the end of a tool development project, for the purpose of assuring that the tool life cycle processes are complete, tool life cycle data is complete, and the tool executable object code is controlled and can be regenerated.

Tool development environment – The environment where the tool is developed, including the methods and tools to be used for the activities of each tool life cycle process.

Tool life cycle – An ordered collection of processes determined by an organization to be sufficient and adequate to produce a tool product.

Tool life cycle environment – See definition for "Tool development environment".

Tool operational environment – The environment and life cycle process context in which the tool is used. It includes workstation, operating system, and external dependencies, for example interfaces to other tools and manual process(es).

Tool Operational Requirements – The Tool Operational Requirements can be viewed as the "system requirements" for a tool. Tool Operational Requirements are from the perspective of the user and may not provide all requirements necessary to develop a tool.

Tool qualification – The process necessary to obtain certification credit for a software tool within the context of a specific system.

Tool Requirements – The Tool Requirements are the requirements used to develop and verify a tool. They include all functional, interface, and safety-related requirements that will be implemented in the tool. Tool Requirements may be presented in multiple levels of requirements.

Tool service history – A contiguous period of time during which the tool is operated within a known environment, and during which successive failures are recorded.

For Personal Use by Jordan Colson, Skyryse

© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Tool Source Code – Code written in source languages, such as assembly language and/or high level language, in a machine-readable form for input to an assembler or a compiler.

Tool verification environment – The environment where the tool in its executable form is verified. The tool verification environment should be representative of the tool operational environment.

Trace Data – Data providing evidence of traceability of development and verification processes' tool life cycle data without implying the production of any particular artifact. Trace Data may show linkages, for example, through the use of naming conventions or through the use of references or pointers either embedded in or external to the tool life cycle data.

Traceability – An association between items, such as between process outputs, between an output and its originating process, or between a requirement and its implementation.

Transition criteria – The minimum conditions, as defined by the tool planning process, to be satisfied to enter a process.

Unintended function – A function included in tool software that is not part of the requirements or design for the tool. Tools may contain functions that are not utilized in all applications. Tool Operational Requirements define the functions used for a given application, whereas Tool Requirements define all functions implemented by the tool. Unintended functions are functions that are not defined in the Tool Requirements.

Validation – The process of determining that the requirements are the correct requirements and that they are complete.

Verification – The evaluation of the outputs of a process to ensure correctness and consistency with respect to the inputs and standards provided to that process.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

This Page Intentionally Left Blank

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

APPENDIX A MEMBERSHIP LIST

EXECUTIVE COMMITTEE MEMBERS

Jim Krodel, Pratt & Whitney	SC-205 Chair
Gérard Ladier, Airbus/Aerospace Valley	WG-71 Chair
Mike DeWalt, Certification Services, Inc./FAA	SC-205 Secretary (until March 2008)
Leslie Alford, Boeing Company	SC-205 Secretary (from March 2008)
Ross Hannan, Sigma Associates (Aerospace)	WG-71 Secretary
Barbara Lingberg, FAA	FAA Representative/CAST Chair
Jean-Luc Delamaide, EASA	EASA Representative
John Coleman, Dawson Consulting	Sub-group Liaison
Matt Jaffe, Embry-Riddle Aeronautical University	Web Site Liaison
Todd R. White, L-3 Communications/Qualtech	Collaborative Technology Software Liaison

SUB-GROUP LEADERSHIP

SG-1 – Document Integration

Ron Ashpole, SILVER ATENA	SG-1 Co-chair
Tom Ferrell, Ferrell and Associates Consulting	SG-1 Co-chair (until March 2008)
Marty Gasiorowski, Worldwide Certification Services	SG-1 Co-chair (from March 2008)
Tom Roth, Airborne Software Certification Consulting	SG-1 Secretary

SG-2 – Issues and Rationale

Ross Hannan, Sigma Associates (Aerospace)	SG-2 Co-chair
Mike DeWalt, Certification Services, Inc./FAA	SG-2 Co-chair (until March 2008)
Will Struck, FAA	SG-2 Co-chair (until March 2009)
Fred Moyer, Rockwell Collins	SG-2 Co-chair (from April 2009)
John Angermayer, Mitre	SG-2 Secretary

SG-3 – Tool Qualification

Frédéric Pothon, ACG Solutions	SG-3 Co-chair
Leanna Rierson, Digital Safety Consulting	SG-3 Co-chair
Bernard Dion, Esterel Technologies	SG-3 Co-secretary
Gene Kelly, CertTech	SG-3 Co-secretary (until May 2009)
Mo Piper, Boeing Company	SG-3 Co-secretary (from May 2009)

SG-4 – Model-Based Development and Verification

Pierre Lionne, EADS APSYS	SG-4 Co-chair
Mark Lillis, Goodrich GPECS	SG-4 Co-chair
Hervé Delseny, Airbus	SG-4 Co-chair
Martha Blankenberger, Rolls-Royce	SG-4 Secretary

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc.
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

SG-5 – Object-Oriented Technology

Peter Heller, Airbus Operations GmbH	SG-5 Co-chair (until February 2009)
Jan-Hendrik Boelens, Eurocopter	SG-5 Co-chair (Feb 2009 to August 2010)
James Hunt, aicas	SG-5 Co-chair (from August 2010)
Jim Chelini, Verocel	SG-5 Co-chair (until Oct 2009)
Greg Millican, Honeywell	SG-5 Co-chair (Oct 2009 to August 2011)
Jim Chelini, Verocel	SG-5 Co-chair (from August 2011)

SG-6 – Formal Methods

Duncan Brown, Aero Engine Controls (Rolls-Royce)	SG-6 Co-chair
Kelly Hayhurst, NASA	SG-6 Co-chair

SG-7 – Special Considerations and CNS/ATM

David Hawken, NATS	SG-7 Co-chair (until June 2010)
Jim Stewart, NATS	SG-7 Co-chair (from June 2010)
Don Heck, Boeing Company	SG-7 Co-chair
Leslie Alford, Boeing Company	SG-7 Secretary (until March 2008)
Marguerite Baier, Honeywell	SG-7 Secretary (from March 2008)

RTCA Representative:

Rudy Ruana	RTCA Inc. (until September 2009)
Ray Glennon	RTCA Inc. (until March 2010)
Hal Moses	RTCA Inc. (until August 2010)
Cyndy Brown	RTCA Inc. (until August 2011)
Hal Moses	RTCA Inc. (from August 2011)

EUROCAE Representative:

Gilbert Amato	EUROCAE (until September 2009)
Roland Mallwitz	EUROCAE (from October 2009)

EDITORIAL COMMITTEE

Leanna Rierson, Digital Safety Consulting	Editorial Committee Chair
Ron Ashpole, SILVER ATENA	Editorial Committee
Alex Ayzenberg, Boeing Company	Editorial Committee
Patty (Bartels) Bath, Esterline AVISTA	Editorial Committee
Dewi Daniels, Verocel	Editorial Committee
Hervé Delseny, Airbus	Editorial Committee
Andrew Elliott, Design Assurance	Editorial Committee
Kelly Hayhurst, NASA	Editorial Committee
Barbara Lingberg, FAA	Editorial Committee
Steven C. Martz, Garmin	Editorial Committee
Steve Morton, TBV Associates	Editorial Committee
Marge Sonnek, Honeywell	Editorial Committee

© 2011 RTCA, Inc.

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

COMMITTEE MEMBERSHIP

Name	Organization
Kyle Achenbach	Rolls-Royce
Dana E. Adkins	Kidde Aerospace
Leslie Alford	Boeing Company
Carlo Amalfitano	Certon Software, Inc
Gilbert Amato	EUROCAE
Peter Amey	Praxis High Integrity Systems
Robert Annis	GE Aviation
Allan Gilmour Anderson	Embraer
Håkan Anderwall	Saab AB
Joseph Angelo	NovAtel Inc, Canada
John Charles Angermayer	Mitre Corp
Ron Ashpole	SILVER ATENA
Alex Ayzenberg	Boeing Company
Marguerite Baier	Honeywell
Fred Barber	Avidyne
Clay Barber	Garmin International
Gerald F. Barofsky	L-3 Communications
Patty (Bartels) Bath	Esterline AVISTA
Brigitte Bauer	Thales
Phillipe Baufreton	SAGEM DS Safran Group
Connie Beane	ENEA Embedded Technology Inc
Bernard Beaudouin	EADS APSYS
Germain Beaulieu	Independent Consultant
Martin Beeby	Seaweed Systems
Scott Beecher	Pratt & Whitney
Haik Biglari	Fairchild Controls
Peter Billing	Aviya Technologies Inc
Denise Black	Embedded Plus Engineering
Brad Blackhurst	Independent Consultant
Craig Bladow	Woodward
Martha Blankenberger	Rolls-Royce
Holger Blasum	SYSGO
Thomas Bleichner	Rohde & Schwarz
Don Bockenfeld	CMC Electronics
Jan-Hendrik Boelens	Eurocopter
Eric Bonnafous	CommunicationSys
Jean-Christophe Bonnet	CEAT
Hugues Bonnin	Cap Gemini
Matteo Bordin	AdaCore
Feliks Bortkiewicz	Boeing
Julien Bourdeau	DND (Canada)
Paul Bousquet	Volpe National Transportation Systems Center
David Bowen	EUROCAE
Elizabeth Brandli	FAA
Andrew Bridge	EASA
Paul Brook	Thales
Daryl Brooke	Universal Avionics Systems Corporation
Cyndy Brown	RTCA, Inc.
Duncan Brown	Aero Engine Controls (Rolls-Royce)

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.
© 2011 RTCA, Inc.

Name	Organization
Thomas Buchberger	Siemens AG
Brett Burgeles	Consultant
Bernard Buscail	Airbus
Bob Busser	Systems and Software Consortium
Christopher Caines	QinetiQ
Cristiano Campos Almeida De Freitas	Embraer
Jean-Louis Camus	Esterel Technologies
Richard Canis	EASA
Yann Carlier	DGAC
Luc Casagrande	EADS Apsys
Mark Chapman	Hamilton Sundstrand
Scott Chapman	FAA
Jim Chelini	Verocel
Daniel Chevallier	Thales
John Chilenski	Boeing Company
Subbiah Chockalingam	HCL Technologies
Chris Clark	Sysgo
Darren Cofer	Rockwell Collins
Keith Coffman	Goodrich
John Coleman	Dawson Consulting
Cyrille Comar	AdaCore
Ray Conrad	Lockheed Martin
Mirko Conrad	The MathWorks, Inc.
Nathalie Corbovianu	DGAC
Ana Costanti	Embraer
Dewi Daniels	Verocel
Eric Danielson	Rockwell Collins
Henri De La Vallée Poussin	SABCA
Michael Deitz	Gentex Corporation
Jean-Luc Delamaide	EASA
Hervé Delseny	Airbus
Patrick Desbiens	Transport Canada
Mike DeWalt	Certification Services, Inc./FAA
Mansur Dewshi	Ultra Electronics Controls
Bernard Dion	Esterel Technologies
Antonio Jose Vitorio Domiciano	Embraer
Kurt Doppelbauer	TTTech
Cheryl Dorsey	Digital Flight
Rick Dorsey	Digital Flight
John Doughty	Garmin International
Vincent Dovydaitis III	Foliage Software Systems, Inc.
Georges Duchein	DGA
Branimir Dulic	Transport Canada
Gilles Dulon	SAGEM DS Safran Group
Paul Dunn	Northrop Grumman Corporation
Andrew Eaton	UK CAA
Brian Eckmann	Universal Avionics Systems Corporation
Vladimir Eliseev	Sukhoi Civil Aircraft Company (SCAC)
Andrew Elliott	Design Assurance
Mike Elliott	Boeing Company
Joao Esteves	Critical Software

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Name	Organization
Rowland Evans	Pratt & Whitney Canada
Louis Fabre	Eurocopter
Martin Fassl	Siemens AG
Michael Fee	Aero Engine Controls (Rolls-Royce)
Tom Ferrell	Ferrell and Associates Consulting
Uma Ferrell	Ferrell and Associates Consulting
Lou Fisk	GE Aviation
Ade Fountain	Penny and Giles
Claude Fournier	Liebherr
Pierre Francine	Thales
Timothy Frey	Honeywell
Stephen J. Fridrick	GE Aviation
Leonard Fulcher	TTTech
Randall Fulton	Seaweed Systems
Francoise Gachet	Dassault-Aviation
Victor Galushkin	GosNIIAS
Marty Gasiorowski	Worldwide Certification Services
Stephanie Gaudan	Thales
Jean-Louis Gebel	Airbus
Dries Geldof	BARCO
Dimitri Gianesini	Airbus
Jim Gibbons	Boeing Company
Dara Gibson	FAA
Greg Gicca	AdaCore
Steven Gitelis	Lumina Engineering
Ian Glazebrook	WS Atkins
Santiago Golmayo	GMV SA
Ben Gorry	British Aerospace Systems
Florian Gouleau	DGA Techniques Aéronautiques
Olivier Graff	Intertechnique - Zodiac
Russell DeLoy Graham	Garmin International
Robert Green	BAE Systems
Mark Grindle	Systems Enginuity
Peter Grossinger	Pilatus Aircraft
Mark Gulick	Solers, Inc.
Pierre Guyot	Dassault Aviation
Ibrahim Habli	University of York
Ross Hannan	Sigma Associates (Aerospace) Limited
Christopher H. Hansen	Rockwell Collins
Wue Hao Wen	Civil Aviation Administration of China (CAAC)
Keith Harrison	HVR Consulting Services Ltd
Bjorn Hasselqvist	Saab AB
Kevin Hathaway	Aero Engine Controls (Goodrich)
David Hawken	NATS
Kelly Hayhurst	NASA
Peter Heath	Securaplane Technologies
Myron Hecht	Aerospace Corporation
Don Heck	Boeing Company
Peter Heller	Airbus Operations GmbH
Barry Hendrix	Lockheed Martin
Michael Hennell	LDRA

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

Name	Organization
Michael Herring	Rockwell Collins
Ruth Hirt	FAA
Kent Hollinger	Mitre Corp
C. Michael Holloway	NASA
Ian Hopkins	Aero Engine Controls (Rolls-Royce)
Gary Horan	FAA
Chris Hote	PolySpace Inc.
Susan Houston	FAA
James Hummell	Embedded Plus
Dr. James J. Hunt	aicas
Rebecca L. Hunt	Boeing Company
Stuart Hutchesson	Aero Engine Controls (Rolls-Royce)
Rex Hyde	Moog Inc. Aircraft Group
Mario Iacobelli	Mannarino Systems
Melissa Isaacs	FAA
Vladimir Istomin	Sukhoi Civil Aircraft Company (SCAC)
Stephen A. Jacklin	NASA
Matt Jaffe	Embry-Riddle Aeronautical University
Marek Jaglarz	Pilatus Aircraft
Myles Jalalian	FAA
Merlin James	Garmin International
Tomas Jansson	Saab AB
Eric Jenn	Thales
Lars Johannknecht	EADS
Rikard Johansson	Saab AB
John Jorgensen	Universal Avionics Systems
Jeffrey Joyce	Critical Systems Labs
Chris Karis	Ensco
Gene Kelly	CertTech
Anne-Cécile Kerbrat	Aeroconseil
Randy Key	FAA
Charles W. Kilgore II	FAA
Wayne King	Honeywell
Daniel Kinney	Boeing Company
Judith Klein	Lockheed Martin
Joachim Klichert	Diehl Avionik Systeme
Jeff Knickerbocker	Sunrise Certification & Consulting, Inc.
John Knight	University of Virginia
Rainer Kollner	Verocel
Andrew Kornecki	Embry-Riddle Aeronautical University
Igor Koverninskiy	Gos NIIAS
Jim Krodel	Pratt & Whitney
Paramesh Kunda	Pratt & Whitney Canada
Sylvie Lacabanne	AIRBUS
Gérard Ladier	Airbus/Aerospace Valley
Ron Lambalot	Boeing Company
Boris Langer	Diehl Aerospace
Susanne Lanzerstorfer	APAC GesmbH
Gilles Laplane	SAGEM DS Safran Group
Jeanne Larsen	Hamilton Sundstrand
Emmanuel Ledinot	Dassault Aviation

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Name	Organization
Stephane Leriche	Thales
Hong Leung	Bell Helicopter Textron
John Lewis	FAA
John Li	Thales
Mark Lillis	Goodrich GPECS
Barbara Lingberg	FAA
Pierre Lionne	EADS APSYS
Hoyt Lougee	Foliage Software Systems
Howard Lowe	GE Aviation
Hauke Luethje	NewTec GmbH
Jonathan Lynch	Honeywell
Françoise Magliozzi	Atos Origin
Veronique Magnier	EASA
Kristine Maine	Aerospace Corporation
Didier Malescot	DSNA/DTI
Varun Malik	Hamilton Sundstrand
Patrick Mana	EUROCONTROL
Joseph Mangan	Coanda Aerospace Software
Ghilaine Martinez	DGA Techniques Aéronautiques
Steven C. Martz	Garmin International
Peter Matthews	Independent Consultant
Frank McCormick	Certification Services Inc
Scott McCoy	Harris Corporation
Thomas McHugh	FAA
William McMinn	Lockheed Martin
Josh McNeil	US Army AMCOM SED
Kevin Meier	Cessna Aircraft Company
Amanda Melles	Bombardier
Marc Meltzer	Belcan Engineering
Steven Miller	Rockwell Collins
Gregory Millican	Honeywell
John Minihan	Resource Group
Martin Momberg	Cassidian Air Systems
Pippa Moore	UK CAA
Emilio Mora-Castro	EASA
Endrich Moritz	Technical University
Robert Morris	CDL Systems Ltd.
Allan Terry Morris	NASA
Steve Morton	TBV Associates
Harold Moses	RTCA, Inc.
Nadir Mostefat	Mannarino Systems
Fred B. Moyer	Rockwell Collins
Robert D. Mumme	Embedded Plus Engineering
Arun Murthi	AERO&SPACE USA
Armen Nahapetian	Teledyne Controls
Gerry Ngu	EASA
Elisabeth Nguyen	Aerospace Corporation
Robert Noel	Mitre Corp
Sven Nordhoff	SQS AG
Paula Obeid	Embedded Plus Engineering
Eric Oberle	Becker Avionics

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.
© 2011 RTCA, Inc.

Name	Organization
Brenda Ocker	FAA
Torsten Ostermeier	Bundeswehr
Frederic Painchaud	Defence Research and Development Canada
Sean Parkinson	Resource Group
Dennis Patrick Penza	AVISTA
Jean-Phillipe Perrot	Turbomeca
Robin Perry	GE Aviation
David Petesch	Hamilton Sundstrand
John Philbin	Northrop Grumman Integrated Systems
Christophe Piala	Thales Avionics
Cyril Picard	EADS APSYS
Francine Pierre	Thales Avionics
Patrick Pierre	Thales Avionics
Gerald Pilj	FAA
Benoit Pinta	Intertechnique - Zodiac
Mo Piper	Boeing Company
Andreas Pistek	ITK Engineering AG
Laurent Plateaux	DGA
Laurent Pomies	Independent Consultant
Jennifer Popovich	Jeppesen Inc.
Clifford Porter	Aircell LLC
Frédéric Pothon	ACG Solutions
Bill Potter	The MathWorks Inc
Sunil Prasad	HCL Technologies, Chennai, India
Paul J. Prisaznuk	ARINC-AEEC
Naim Rahmani	Transport Canada
Angela Rapaccini	ENAC
Lucas Redding	Silver-Atena
David Redman	Aerospace Vehicle Systems Institute (AVSI)
Tammy Reeve	Patmos Engineering Services, Inc.
Guy Renault	SAGEM DS Safran Group
Leanna Rierson	Digital Safety Consulting
George Romanski	Verocel
Cyrille Rosay	EASA
Edward Rosenbloom	Kollsman, Inc
Tom Roth	Airborne Software Certification Consulting
Jamel Rouahi	CEAT
Marielle Roux	Rockwell Collins France
Benedito Massayuki Sakugawa	ANAC Brazil
Almudena Sanchez	GMV SA
Vdot Santhanam	Boeing Company
Laurence Scales	Thales
Deidre Schilling	Hamilton Sundstrand
Ernst Schmidt	Bundeswehr
Peter Schmitt	Universität Karlsruhe
Dr. Achim Schoenhoff	EADS Military Aircraft
Martin Schwarz	TT Technologies
Gabriel Scolan	SAGEM DS Safran Group
Christel Seguin	ONERA
Beatrice Sereno	Teuchos SAFRAN
Phillip L. Shaffer	GE Aviation

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Name	Organization
Jagdish Shah	Parker
Vadim Shapiro	TetraTech/AMT
Jean François Sicard	DGA Techniques Aéronautiques
Marten Sjoestedt	Saab AB
Peter Skaves	FAA
Greg Slater	Rockwell Collins
Claudine Sokoloff	Atos Origin
Marge Sonnek	Honeywell
Guillaume Soudain	EASA
Roger Souter	FAA
Robin L. Sova	FAA
Richard Spencer	FAA
Thomas Sperling	The Mathworks
William StClair	LDRA
Roland Stalford	Galileo Industries Spa
Jerry Stamatopoulous	Aircell LLC
Tom Starnes	Cessna Aircraft Company
Jim Stewart	NATS
Tim Stockton	Certon
Victor Strachan	Northrop-Grumman
John Strasburger	FAA
Margarita Strelnikova	Sukhoi Civil Aircraft Company (SCAC)
Ronald Stroup	FAA
Will Struck	FAA
Wladimir Terzic	SAGEM DS Safran Group
Wolfgang Theurer	C-S SI
Joel Thornton	Tier5 Inc
Mikael Thorvaldsson	KnowIT Technowledge
Bozena Brygida Thrower	Hamilton Sundstrand
Christophe Travers	Dassault Aviation
Fay Trowbridge	Honeywell
Nick Tudor	Tudor Associates
Silpa Uppalapati	FAA
Marie-Line Valentin	Airbus
Jozef Van Baal	Civil Aviation Authorities Netherlands
John Van Leeuwen	Sikorsky Aircraft
Aulis Viik	NAV Canada
Bertrand Voisin	Dassault Aviation
Katherine Volk	L-3 Communications
Dennis Wallace	FAA
Andy Wallington	Bell Helicopter
Yunming Wang	Esterel Technologies
Don Ward	AVSI
Steve Ward	Rockwell Collins
Patricia Warner	Software Engineering
Michael Warren	Rockwell Collins
Rob Weaver	NATS
Yu Wei	CAA China
Terri Weinstein	Parker Hannifin
Marcus Weiskirchner	EADS Military Aircraft
Daniel Weisz	Sandel Avionics, Inc.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

Name	Organization
Rich Wendlandt	Quantum3D
Michael Whalen	Rockwell Collins
Paul Whiston	High Integrity Solutions Ltd
Todd R. White	L-3 Communications/Qualtech
Virginie Wiels	ONERA
ElRoy Wiens	Cessna Aircraft Company
Terrance Williamson	Jeppesen Inc.
Graham Wisdom	BAE Systems
Patricia Wojnarowski	Boeing Commercial Airplanes
Joerg Wolfrum	Diehl Aerospace
Kurt Woodham	NASA
Cai Yong	CAAC (Civil Aviation Administration of China)
Edward Yoon	Curtiss-Wright Controls, Inc
Robert Young	Rolls-Royce
William Yu	CAAC China
Erhan Yuceer	Savunma Teknolojileri Muhendislik ve Ticaret
Uli Zanker	Liebherr

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

APPENDIX B EXAMPLE OF DETERMINATION OF APPLICABLE TOOL QUALIFICATION LEVELS

This information is extracted from DO-278A, applicable to ground-based software, such as CNS and ATM software. It serves as an example of how one domain added criteria to their guidance in order to recognize this document.

Determining if Tool Qualification is Needed

Qualification of a tool is needed when processes of this document are eliminated, reduced or automated by the use of a software tool without its output being verified as specified in section 6.

The purpose of the tool qualification process is to ensure that the tool provides confidence at least equivalent to that of the process(es) eliminated, reduced or automated.

The tool qualification process may be applied to a single tool, a collection of tools, or one or more functions within a tool. For a tool with multiple functions, if protection between tool functions can be demonstrated, only those functions that are used to eliminate, reduce or automate software life cycle processes, and whose outputs are not verified, need be qualified. Protection is the use of a mechanism to ensure that a tool function cannot adversely impact another tool function.

A tool is qualified only for use on a specific system where the intention to use the tool is stated in the Plan for Software Aspects of Approval that supports the system. If a tool is used for another system, it should be re-qualified within the context of that system.

Determining the Tool Qualification Level (TQL)

If tool qualification is needed, the impact of the tool use in the software life cycle process should be assessed in order to determine its tool qualification level. The following criteria should be used to determine the impact of the tool:

a. Criteria1: A tool whose output is part of the resulting software and thus could insert an error.

b. Criteria 2: A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of:

    1. Verification process(es) other than that automated by the tool, or
    2. Development process(es) that could have an impact on the CNS/ATM software.

c. Criteria 3: A tool that, within the scope of its intended use, could fail to detect an error.

If the tool eliminates, reduces, or automates processes in this document and its output is not verified as specified in section 6, the appropriate TQL is as shown in Table 12-1. Five levels of tool qualification are identified based on the tool use and its potential impact in the software life cycle processes. TQL-1 is the most rigorous level and TQL-5 is the least rigorous. When assessing the impact of a given tool, the criteria should be considered

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

sequentially from criteria 1 to criteria 3. The tool qualification level should be coordinated with the approval authority as early as possible.

Table 12-1 Tool Qualification Level Determination

Assurance Level	Criteria		
	1	2	3
AL1	TQL-1	TQL-4	TQL-5
AL2	TQL-2	TQL-4	TQL-5
AL3	TQL-3	TQL-5	TQL-5
AL4	TQL-4	TQL-5	TQL-5
AL5	TQL-4	TQL-5	TQL-5

Tool Qualification Process

The objectives, activities, guidance, and life cycle data required for each Tool Qualification Level are described in DO-330, "Software Tool Qualification Considerations."

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

APPENDIX C FREQUENTLY ASKED QUESTIONS RELATED TO TOOL QUALIFICATION FOR ALL DOMAINS

1.0 APPENDIX C INTRODUCTION

This appendix provides clarification of tool qualification issues through a series of Frequently Asked Questions (FAQs). FAQs vary in length – some are short and concise, whereas others are longer and provide more detail. FAQs do not provide guidance but merely provide clarification.

The FAQs in this appendix apply to tools in any domain. Domain-specific FAQs for tools are included in other Appendices or may even be in the domain-specific documentation itself.

1.1 FAQ C.1: What Does "Protection" Mean for Tools and What Are Some Means to Achieve It?

Reference: This document: Sections 5.2.2.2, 6.1.3.3, 10.2.2, and 11.1

Keywords: tool qualification; protection; multi-function tool

Answer:

When differing levels of qualification are applied to a multi-function tool or a collection of tools, protection between the different functions is required. The glossary defines protection as: "The use of a mechanism to ensure that a tool function cannot adversely impact another tool function." In practical terms, "protection" between tool functions may be any mechanism in the tool's architecture, design, or implementation that prevents adverse interference between different functions of the tool.

Protection may be achieved using numerous techniques such as those listed below:

- The implementation of the two functions may be partitioned spatially and temporally.
- Similarly, two functions that perform different operations on the same input data stream may be protected from each other by creating a second output data stream for the second function. Essentially, the common processing of the input to the functions is branched into dual streams in order to disallow the two functions from interfering with each other's correct operation. This technique employs a functional partitioning strategy, and not necessarily a spatial or temporal partition strategy.
- Qualified tool functions may be protected from other functions by using functional deactivation techniques. Using a control input to the tool, the tool is restricted from performing the second function, thereby protecting the first from any adverse impact. The method of deactivating the second function should be subject to appropriate verification as part of the tool qualification.

This list of protection mechanisms is by no means complete, but does represent the intent of the term "protection". Any specific protection mechanism employed in a tool to be qualified needs to be reflected in the Tool Requirements and/or Tool Operational Requirements, and agreement with the certification authority on the efficacy and appropriateness of the protection mechanism will need to be reached.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

1.2 FAQ C.2: What Are External Components and How Does One Assess Their Correctness?

Reference: This document: Sections 5.2.2, 6.1.3.3, 6.1.3.4, 6.1.3.5, 6.1.4.3, 10.1.2, and 10.3.2

Keywords: tool qualification; structural coverage; external component

Answer:

Structural coverage analysis is typically performed at the Source Code level. For external components this analysis may be difficult or impractical. Consequently, additional verification is required for TQL-1 on the external components to assess the correctness of their functionality.

1.2.1 What Are the External Components of a Tool?

There may be functionality in the tool that relies on software components, such as the compiler run-time library (providing functions such as formatting character strings, math operations, etc.) or the operating system primitives. These components are outside the control of the tool developers, however, an acceptable level of confidence should be provided in the correct functionality of these external components

The following components should not be considered as "external", and the approach described in section 1.2.2 of this appendix is not applicable to them. These components are:

- Components developed for limited use or specifically to support the tool. These components need to be considered as under the tool developer's control, and structural coverage analysis would apply.
- Libraries of functions that are used in tool operation (for example, code that implements basic symbols of a model). These libraries are subject to structural coverage analysis as part of the tool qualification effort, using the applicable TQL.
- Libraries used or incorporated in the tool's output that becomes part of the resultant software. These used library components need to be verified as part of the software life cycle processes. This is also applicable to libraries provided by a tool vendor.

1.2.2 Application of Structural Coverage Analysis for Tool External Components

Application of structural coverage on the external components may be difficult or impractical for the following reasons:

- In the case of a compiler run-time library, the compiler user's manual or language reference manual may have information that could be used as requirements. However, the Source Code may be difficult or costly to obtain, making structural coverage analysis difficult or impractical.
- In the case of the low-level operating system functions utilized by the tool, neither requirements nor Source Code would typically be available, making this area impractical to assess for structural coverage.

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

If structural coverage analysis were to be performed on this type of software, application of structural coverage analysis resolution may result in the following limitations:

- Shortcomings in requirements-based test cases or procedures may not be pertinent to the functionality of the tool. For functions not utilized by the tool, there is little safety benefit in adding test cases to exercise these functions.
- Inadequacies in the software requirements cannot be addressed, because the requirements are limited or unavailable.
- Dead code cannot be removed, because typically the Source Code is not available, nor would it be practical to modify the Source Code.
- Deactivated code and unused function cannot be fully addressed due to the lack of requirements and/or Source Code.

The correct functionality of the external components is assessed through:

- Their identification during the design process.
- The verification of the correctness of their identification and of their interfaces during the design review.
- An additional requirements-based tests coverage analysis. This analysis needs to ensure that the tests exercise the interface and the functionality of each function of the external components utilized by the tool.

1.3 FAQ C.3: How Can One Maximize Reusability of Tool Qualification Data?

Reference: This document: Section 11

Keywords: tool qualification; reusability

Answer:

Maximizing the reusability of tool qualification data requires some forethought and planning during the initial tool qualification effort. The tool data need to be packaged for reuse in multiple projects. Each qualifying project also needs to ensure that the data is reusable for their given project. The suggestions below outline a general approach to enhancing reusability, but should not be considered as exhaustive.

a. Qualification data should be packaged to facilitate reusability. Section 11.3 (Qualifying COTS Tools) discusses the roles and responsibilities for satisfying the tool qualification objectives. To qualify a previously qualified tool without any change in the context of the tool use, only the activities under the responsibility of the user are assessed and may need to be repeated.

b. Objectives assigned to the tool developer's responsibility are considered to be user-independent. It is suggested that:

   - User-independent data be packaged separately from user-dependent data.
   - User-dependent data reference the user-independent data.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

This concept is also applicable for TQL-5 tools, which are not specifically required to have qualification data separate from the software certification data. Although user-independent data is minimized in the qualification of TQL-5 tools, an independent approach to the packaging of the qualification data facilitates the direct reuse of the data, without requiring its reproduction for each subsequent qualification.

c. The tool's user-independent qualification documentation needs to identify all of the functionality anticipated to be qualified by any of the projects. The user-dependent qualification data necessarily identifies the specific functions to be qualified and references the supporting user-independent data.

d. Special-purpose qualification test data and analyses need to be created, where possible, to specifically verify the functionality being qualified. This serves two purposes: First, it clarifies the relationship of the test case or analysis to the requirement(s) being verified. Second, it decouples the verification data from the initial qualifying project. Care needs to be taken to ensure that the verification suite created for the tool is representative of the tool use by the qualifying project(s).

e. Several things need to be considered when planning qualification of the tool on multiple projects. For example, if a tool's functions are coupled in some fashion such that qualification of one function necessarily requires the qualification of a related function, then this coupling needs to be identified in the TQP. Similarly, any restrictions on inputs to the tool need to be included in the TAS. Of particular importance is the clear identification in the TAS of any project or process-related safeguards necessary to enable the qualified use of the tool.

f. At the project level, the PSAC needs to identify any differences between the tool functions to be qualified as compared to the available tool-specific qualification data. The PSAC or project-level TQP needs to identify the specific user activities to be performed to complete the tool qualification within the project's context. Similarly, the multiple-project special considerations identified in the tool's qualification data need to be described with respect to the manner in which the project will accommodate them. Each project needs to describe how it complies with the constraints identified by the tool's qualification data.

g. If a project has multiple tools to be qualified, it may be advisable to create a project TQP and TAS, where the tool-related qualification data specific to the project can be captured for reuse on further certification efforts by the project.

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

APPENDIX D FREQUENTLY ASKED QUESTIONS RELATED TO TOOL QUALIFICATION FOR AIRBORNE SOFTWARE AND CNS/ATM SOFTWARE DOMAINS

1.0 APPENDIX D INTRODUCTION

This appendix provides clarification of tool qualification issues for the DO-178C airborne software domain and the DO-278A CNS/ATM ground-based software domain through a series of FAQs. FAQs vary in length – some are short and concise, whereas others are longer and provide more detail. FAQs do not provide guidance; they only provide clarification.

1.1 FAQ D.1: Why Are the Terms "Verification Tool" and "Development Tool" Not Used to Describe Tools that May Be Qualified?

Reference: DO-178C/DO-278A: Section 12.2

Keywords: DO-178B; tool qualification; tool criteria; verification tool; development tool

Answer:

Since the publishing of DO-178B in December 1992, the use of tools in the development, verification, and management of software has proliferated widely. Not only have new tools been introduced, but new types of tools that perform activities that do not fit cleanly under the "verification tool" and "development tool" categories have become common throughout the industry. The diversity of tool-performed activities is expected to continue throughout the foreseeable future.

Given this diversity, it becomes clear that the words "development" and "verification" are too heavily linked with a functional view of the tool and can be at the same time misleading and overly restrictive. This confusion is created, in part, by the common use of the terms "development" and "verification" to describe particular phases of the process used to create software.

It was perceived that the term "development tool" could be interpreted to include some tools used in the development of software whose output is not integrated into the airborne (or CNS/ATM) software. Similarly, the term "verification tool" could be interpreted to exclude some tools which could be beneficially qualified because the tool does not perform a "verification activity". Additionally, there are tools that do not cleanly fall into the misinterpreted "development tool" and "verification tool" categories that should be qualified but are not due to the misunderstood labels.

Thus, there was a clear need to address the problem by asking the essential question: What is the effect of anomalous behavior by the tool on the airborne (or CNS/ATM) software? This was the question already asked in DO-178B to determine the "development" or "verification" tool category.

The answer to this question, combined with the software level of the airborne (or CNS/ATM) software determines the TQL as described in section 12.2 of DO-178C/DO-278A, without assigning a name to the "kind" of tool being qualified.

This approach accommodates tools that do not easily fit into these named categories and allows for the future expansion of the definition of tools that may be qualified.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.
© 2011 RTCA, Inc.

1.2 FAQ D.2: Can TQL Be Reduced?

Reference: DO-178C/DO-278A: Section 12.2

Keywords: tool qualification; tool qualification level; TQL; architectural mitigation

Answer:

DO-178C/DO-278A section 12.2 defines how to assign TQL. For TQL-1 through TQL-4, the assigned TQL can be reduced when agreed upon by the certification authority. Architectural mitigation, monitoring, independent evaluations, etc. may affect the overall level required for a tool qualification. The reduction in TQL needs to be closely coordinated with the certification authority and documented in the PSAC. (Note: For DO-278A users, "certification authority" is equivalent to "approval authority", and a PSAA is used instead of a PSAC.)

Reduction in a tool's qualification level needs to be based largely upon the significance of the DO-178C/DO-278A activity to be eliminated, reduced, or automated, with respect to the entire suite of verification activities. This significance is a function of:

- The type of DO-178C/DO-278A activity to be eliminated, reduced, or automated. For example, the automation of the verification of compliance of Source Code to some presentation rules of the Coding Standard is less significant than the automation of the verification of compliance of the Source Code to safety rules of the Coding Standard.
- The likelihood that other activities would have detected the same error(s).

To reduce a tool's qualification level, the reduction needs to be justified by performing a tool use and impact analysis. This analysis needs to evaluate the overall use of the tool in the development process and its impact on the software being produced. The analysis needs to be included in the PSAC (or PSAA for DO-278A users).

1.3 FAQ D.3: When Do Target Computer Emulators or Simulators Need to Be Qualified?

Reference: DO-178C/DO-278A: Sections 4.4.3, 6.4.1, 6.4.3, 12.2; and Glossary

Keywords: tool qualification; emulator; simulator; test environment

Answer:

This FAQ is restricted to emulation and simulation of the target computer (including the processor and memory) for testing Executable Object Code. Other uses of emulator or simulator to eliminate, reduce, or automate satisfaction of objectives are out of scope for this discussion.

1.3.1 DO-178C/DO-278A Background and Guidance

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

DO-178C/DO-278A section 6.4 defines three types of testing: hardware/software integration testing, software integration testing, and low-level testing. The DO-178C/DO-278A guidance related to verification environments which would include emulation and simulation environments is provided in section 6.4.1.

DO-178C/DO-278A acknowledges that more than one test environment may be needed to satisfy the objectives for software testing. DO-178C/DO-278A states in section 6.4.3 that with the exception of hardware/software integration testing, the methods do not prescribe a specific test environment or strategy. Hardware/software integration testing is clearly defined as needing to be done on the target computer environment with a focus on requirements-based testing of the high-level functionality, the hardware/software interfaces, and the error sources associated with the software operating within the target computer environment, as defined in DO-178C/DO-278A section 6.4.1.a.

1.3.2 Use of an Emulator or Simulator

When emulator or simulator tools are used in testing, they are considered part of the test environment. Specific tests executed in this environment need to be identified and distinguished from tests performed on the target computer. Test selection for execution in the emulator or simulator environment relies on analysis of the representation of the emulator or simulator environment for the target computer environment.

There are two main uses of emulation and simulation:

a. Emulation and simulation used in framework of hardware/software integration tests. In this case, the emulator or simulator environments replace the target computer for tests that need to be executed on target computer environment as defined in DO-178C/DO-278A section 6.4.1.a. The use of an emulator or simulator in this manner can be seen as eliminating, reducing, or automating the satisfaction of target computer compatibility objectives. In this case, tests executed in the emulator or simulator environment may fail to detect an error if the emulator or simulator environment is not representative of the target computer environment. Therefore, the applicant needs to provide evidence that the emulator or simulator environment is equivalent to the target computer environment for aspects affecting the specific hardware/software integration tests to be performed on the emulated or simulated environment. Discrepancies between the emulator or simulator environment and the target computer environment that do not affect the specific tests to be performed in the emulator or simulator environment need to be documented.

b. Emulation and simulation not used in framework of hardware/software integration tests. In this case, the emulator or simulator tools are part of the verification environment and should be configuration controlled. The satisfaction of target computer compatibility objectives is not considered to be eliminated, reduced, or automated. However, the emulator or simulator environment is subject to DO-178C/DO-278A section 4.4.3.b.

1.3.3 Demonstration of Equivalence

The differences between the target computer environment and the emulator or simulator environment need to be considered in regard to the ability of tests conducted in the emulator/simulator environment to detect errors typically revealed by the target computer environment testing (see DO-178C/DO-278A section 6.4.1.a) and verify functionalities of the tested software. This could be achieved by analysis (as described in DO-178C/DO-

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

278A section 4.4.3.b) or by emulator or simulator qualification which includes this analysis.

1.3.4 Approach for Qualifying Emulator/Simulator

Tool qualification may be used as a process for demonstrating equivalence for tests to be executed on an emulator or simulator. In this case, the emulator or simulator falls under criteria 3. To satisfy TQL-5 objectives, the TOR and the corresponding tool operational verification and validation process need to demonstrate the equivalence of the tests executed on the emulator or simulator environment as compared to the target computer environment.

1.4 FAQ D.4: What Credit Can Be Granted for Tools Previously Qualified Using DO-178B/DO-278?

Reference: This document: Section 11.2; DO-178C/DO-278A: Section 12.2

Keywords: DO-178B; DO-278; tool qualification; previously qualified tool; legacy tool

Answer:

The regulatory basis for the qualification of a given tool is tied to the certification basis of the project using the tool. A tool used on a project being developed to DO-178C/DO-278A needs to be qualified per the guidance of DO-178C/DO-278A and this document. For tools that have been previously qualified under the guidance of DO-178B/DO-278, additional factors will need to be considered when specifying the qualification basis for the tool, if credit for the previous qualification is desired.

Tools which were previously qualified under DO-178B/DO-278 may be able to be granted certification/approval credit by agreement with the certification/approval authority for use on projects under the guidance of DO-178C/DO-278A. The guidance of the section 11.2 of this document needs to be considered. The amount of credit granted will depend on several factors, including but not limited to:

- The previously qualified tool's type, level, and equivalent DO-178C/DO-278A TQL as identified in the table below.

Table D-1 DO-178B/DO-278 and DO-178C/DO-278A TQL Correlation

DO-178B/DO-278 Tool Qualification Type	DO-178B Software Level	DO-278A Assurance Level	DO-178C/DO-278A TQL
Development	A	AL1	TQL-1
Development	B	AL2	TQL-2
Development	C	AL3	TQL-3
Development	N/A	AL4	TQL-4
Development	D	AL5	TQL-4
Verification	All	All	TQL-4 or TQL-5

- Tool qualification criteria determination. For a tool qualified as a verification tool under DO-178B/DO-278, depending on the credit claimed for the use of the tool, criteria 2 or 3 apply. The applicable TQL will be TQL-4 or TQL-5, as defined in DO-178C/DO-278A section 12.2.

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

- Objective similarity. Whilst the classifications are equivalent, the evidence required to satisfy a particular objective may differ from DO-178B/DO-278 to DO-178C/DO-278A. That is, the objectives (whilst similar) are different (both in quantity and detail). The difference between similar objectives needs to be analyzed and deficiencies addressed. See section 11.2 of this document.

- Objective differences. For any DO-178B/DO-278 objective that has been previously satisfied, there may be an equivalent or similar objective for which credit is available. In the case where a new objective exists, new qualification evidence will need to be provided. See section 11.2 of this document.

The applicant needs to document these factors in their PSAC (or PSAA for DO-278A users) and clearly state the certification/approval credit sought. If it is determined that changes are required to the tool qualification data, the agreement to use either DO-178B/DO-278 or DO-178C/DO-278A compliant tool qualification processes needs to be documented in both the PSAC (or PSAA for DO-278A users) and TQP (as applicable).

1.5 FAQ D.5: What is the Rationale for Tool Qualification Criteria Definition?

Reference: DO-178C/DO-278A: Section 12.2

Keywords: DO-178B; DO-278; tool qualification; previously qualified tool; legacy tool

Answer:

1.5.1 Background

DO-178B/DO-278 addresses two categories of tools: The development tool (a tool that could insert an error) and the verification tool (a tool that could fail to detect an error).

In practice, the names of the tool categories have been inconsistent with respect to the actual tool functionality. As a consequence, some tools may not have been properly addressed in the tool qualification process. Additionally, it has been observed that not all verification tools have the same potential impact on safety. Even if they may not introduce an error in the resulting software, the consequence of the fact that they might only fail to detect an error may vary, depending of the use of the tool in the software life cycle process and the credit claimed.

It is anticipated that the use of formal methods, model-based development, and other tool-intensive methodologies will increase in the future. The use of tools in these methodologies, combined with alleviation of activities in the software life cycle process needs to be properly addressed to ensure safety.

The increased risk that these tool-intensive methodologies brings leads to the conclusion that an intermediate level of tool category between the two DO-178B/DO-278 tool categories is needed in order to address all tool types and to define the appropriate TQLs.

This FAQ explains the rationale behind the tool criteria presented in section 12.2 of DO-178C/DO-278A.

1.5.2 Approach to Define the Applicable TQL

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

DO-178C/DO-278A does not use the terms "development tool" or "verification tool". Instead, it focuses on the impact of a tool error on the resulting software rather than tool category name. The TQL is based on several criteria. Those criteria are related to the impact of a tool in the software life cycle processes and are combined with the resulting software level to define the applicable TQL. The objectives applicable to each TQL are then defined in the Annex A tables of this document.

1.5.3 Tool Criteria Definition

1.5.3.1 Comparison between DO-178B/DO-278 Tool Categories and DO-178C/DO-278A Tool Qualification Criteria

Table D-2 shows the comparison between DO-178B/DO-278 tool categories and DO-178C/DO-278A tool qualification criteria. Criteria should be assessed in order; for example, for a given tool, the criteria 2 needs to be considered after criteria 1 but before criteria 3.

Table D-2 DO-178B/DO-278 Tool Categories and DO-178C/DO-278A Tool Qualification Criteria Comparison

DO-178B/DO-278 Tool Category and Definition	DO-178C/DO-278A Tool Qualification Criteria and Definition
Development tools: Tools whose output is part of airborne (or CNS/ATM) software and thus can introduce errors.	Criteria 1: A tool whose output is part of the resulting software and thus could insert an error
Verification tools: Tools that cannot introduce errors, but may fail to detect them.	Criteria 2: A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of: • Verification process(es) other than that automated by the tool, or • Development process(es) that could have an impact on the airborne (or CNS/ATM) software. Criteria 3: A tool that, within the scope of its intended use, could fail to detect an error.

1.5.3.2 Rationale for Introducing Criteria 2

Traditionally, the verification activities (review, analysis, and test with or without a tool) are performed on a set of life cycle data in order to satisfy a verification objective on this data.

The problem arises when, based on the confidence of a given verification activity, some alleviation is claimed for other objectives or activities that are not the direct purpose of that verification activity.

These alleviations are not allowed with manual verification activities. One of the principles of DO-178B/DO-278 (and that remains in DO-178C/DO-278A) is the

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

overlapping software verification process activities that are applied in order to prevent the insertion of errors or to increase the capabilities to detect errors. That is, multiple verification objectives are applied to identify and remove errors. The higher the software level, the more objectives apply.

However, when the verification is automated with the use of a tool, the applicant may claim that all errors for a specific category are detected by the tool. In this case, some mechanism inserted in the software (during the requirements, design, or coding phases) or some other verification activities are claimed to be redundant and useless. New methods or technologies, such as model-based development and formal methods, are heavily supported by tools. These new methodologies have been considered when documenting the approach in DO-178C/DO-278A and in this document.

When applying the DO-178B/DO-278 tool qualification guidance there was no difference for qualifying tools that could only "fail to detect an error" when used only to automate a verification objective and those for which confidence is claimed to alleviate other objectives or activities.

These additional uses of tools raise some concerns about the rigor required for qualification. For example:

- From a safety perspective, the impact of these tools on the final software can vary.
- The required confidence may not be able to be assessed by considering only the Tool Operational Requirements.
- The maturity and soundness of a mathematical theory and its implementation may be necessary to provide the required confidence.

These concerns combined with the growing use of tools and software development methodologies led to the addition of criteria 2 tools in DO-178C/DO-278A section 12.2. The following section provides some examples of criteria 1, 2, and 3 tools.

1.5.3.3 Considerations about Criteria Selection and Examples

The following considerations and examples are provided to aid in understanding and applying the tool criteria in DO-178C/DO-278A. These limited examples are intended to help with application of the criteria definitions in DO-178C/DO-278A. Applicants need to consider the specificities of their tool use context.

1.5.3.3.1 Application of Criteria 1, 2, and 3

Criteria 1 is applied to the tools that automatically produce a part of the outputs of one of the software development processes, regardless of the input and output format are. This criteria encompasses the tools that transform a higher level of requirements to a lower requirement level (or same level but in a different formalism), to Source Code, to data files, to configuration files, or to Executable Object Code. However, in case of Executable Object Code production, section 4.4.2 of DO-178C/DO-278A also needs to be considered. Tools used in the scope of the production of the outputs of the software development processes, that do not transform the data but that could inject an error in this data also needs to be considered.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

Criteria 2 and Criteria 3 are applied to all tools that verify or analyze software life cycle data, compute software characteristics, etc. Application of one of these two criteria differs based on the certification/approval credit claimed by the applicant.

a. If the certification/approval credit claim is only for the objective directly satisfied by the activity performed by the tool, criteria 3 is applied.

b. An alternative for the applicant is to claim that other objectives are also satisfied or partially satisfied through the use of the tool. In this case, criteria 2 applies.

1.5.3.3.2 Examples of Criteria 2 or 3 Determination

Below are some examples of tools where criteria 2 or 3 may apply.

a. Example 1: A proof tool may be used to automate some verification of Source Code. Criteria 3 could be applied based on this tool's use and credit claimed. However, if the applicant claims that testing activity to detect a class of error becomes unnecessary based on the tool detecting the related class of error, then the criteria 2 becomes applicable. In this case, it corresponds to "a reduction of software verification process(es) other than that automated by the tool."

b. Example 2: A static code analyzer may be used to automate some verification of Source Code review. Criteria 3 could be applied based on this tool's use and credit claimed. However, if the applicant claims to not include some specific mechanisms in the resulting software in order to detect and treat the possible overflow, and run-time errors based on the confidence on the tool, then the criteria 2 becomes applicable. In this case, it corresponds to "a reduction of software development process(es)."

1.5.3.3.3 Multilayered Safety Approach and Tool Qualification Criteria

In some cases, verification activities in DO-178C/DO-278A use a multi-layered safety approach. For example, a non-compliance of Source Code to the low-level requirements may be detected by Source Code review (DO-178C/DO-278A section 6.3.4.a) or by requirements-based tests (DO-178C/DO-278A section 6.4.2). Structural coverage analysis helps in detecting lack of requirements or tests cases. This multi-layered safety approach is made necessary since exhaustive verification may not be possible.

But a Criteria 3 tool that complies with the qualification objectives of this document, gives a consistent verification result so that another process or activity to satisfy the related objective may no longer be needed. This is enforced by the combination of the two following objectives:

- "Tool adequacy is ensured in the Tool Operational Requirements."
- "Tool operation complies with its Tool Operational Requirements."

Therefore, using another process or activity to satisfy the related objective may no longer be needed.

For Criteria 3 tools the multi-layered approach of DO-178C/DO-278A is not compromised if the qualified tool is used to directly eliminate or reduce process(es) of DO-178C/DO-278A within the scope of its intended use. When a tool is also used to eliminate or reduce another process(es), the multi-layered safety approach is not

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

compromised with respect to the previous considerations if enforced by the application of Criteria 2 on this tool. When qualified for use on software with a software level A or B (AL1 or AL2 for DO-278A users), Criteria 2 application leads to a higher TQL than that required of a tool whose intended use meets only Criteria 3.

1.5.3.3.4 TQLs

The applicable TQL is based on the criteria assessment and the software/assurance level. Table D-3 summarizes the TQLs. The text below briefly explains how the new criteria in DO-178C/DO-278A relates to the previous criteria in DO-178B/DO-278.

Table D-3 TQL Summary

DO-178C Software Level	DO-278A Assurance Level	Criteria		
		1	2	3
A	AL1	TQL-1	TQL-4	TQL-5
B	AL2	TQL-2	TQL-4	TQL-5
C	AL3	TQL-3	TQL-5	TQL-5
N/A	AL4	TQL-4	TQL-5	TQL-5
D	AL5	TQL-4	TQL-5	TQL-5

a. Criteria 1: The TQL applicable for criteria 1 is the replacement for the development tool in DO-178B/DO-278.

b. Criteria 2: As explained above, criteria 2 is new for DO-178C/DO-278A and is intended to address the expansion of tool use in new methodologies. Criteria 2 basically requires an increased level of rigor over DO-178B/DO-278 criteria for tools used on software level A and B (AL1 and AL2 for DO-278A users) in order to increase the confidence in the use of the tool (that is, TQL-4 instead of TQL-5). TQL-4 requires that the Tool Requirements data describe all functionality implemented in the tool and provide additional detail about the tool architecture. TQL-4 also requires verification of the compliance of the tool with Tool Requirements. TQL-4 objectives are considered as a minimum to claim confidence in the use of the tool. But the purpose of applying TQL-4 for software level A or B (AL1 or AL2 for DO-278A users) is not to prevent the use of this kind of tool. The following approaches may be considered for tool use:

   - In case of deficiencies in the tool life cycle data needed to qualify the tool at TQL-4, the applicant may still use the tool and qualify it at TQL-5; however, other certification/approval credit is limited to the verification objectives of the data under verification.

   - In case of COTS, if the data life cycle is not provided by the tool supplier to qualify the tool at level TQL-4, section 11 of this document allows an applicant to augment the data in order to satisfy the objectives for the applicable TQL.

   - Other qualification approaches may be possible (see section 11).

c. Criteria 3: The level TQL-5 is the replacement for verification tool in DO-178B/DO-278.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

1.6 FAQ D.6: What "Verification Tool" Qualification Improvements Were Made in DO-178C/DO-278A?

Reference: DO-178C/DO-278A: Section 12.2

Keywords: DO-178B; tool qualification; verification tool; TQL-5; determinism

Answer:

1.6.1 Qualification Criteria for Verification Tools in DO-178B

In DO-178B the guidance applicable to software verification tools is summarized in section 12.2.2:

> "The qualification criteria for software verification tools should be achieved by demonstration that the tool complies with its Tool Operational Requirements under normal operational conditions."

However, DO-178B does not provide details on the content of Tool Operational Requirements, clearly define the development of qualification data for verification tools, nor identify applicable objectives for the tool qualification process.

DO-178B also indicates that for a verification tool:

- The tool qualification data need to be controlled as CC2 (DO-178B section 12.2.3.b).
- TQP and Tool Accomplishment Summary are not required (only input into PSAC and SAS is required) (DO-178B section 12.2.4).
- Generalized objectives of configuration management and quality assurance are applied (since the generalized objectives are not detailed in DO-178B, they are inconsistently applied) (DO-178B section 12.2.c).

1.6.2 Clarifications and Improvements for TQL-5

This section summarizes improvements made in this document for tools at TQL-5. These improvements are intended to clarify items from DO-178B (and hence DO-278).

1.6.2.1 Scope of the qualification

Section 5 identifies the tool development process, which is composed of:

a. The Tool Operational Requirements definition process.

b. The tool development process.

c. The tool operational integration process.

The guidance about the tool development process (item b. above) is not applicable to TQL-5. So, TQL-5 qualification remains a product-oriented approach. The two other processes (items a. and c. above) are defined through specific objectives. The objectives

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

applicable to processes a. and c. above and to the related verification processes are summarized in Table T-0 in Annex A of this document.

The main body of this document provides a complete set of objectives and guidance for all TQLs. Annex A provides a summary of all these objectives with their applicability by TQL. TQL-5 objectives are a subset of the overall set of objectives.

1.6.2.2 Determinism

In DO-178B section 12 the following statement is made: "Only deterministic tools may be qualified, that is, tools which produce the same output for the same input data when operating in the same environment." The use of the term "deterministic" has a specific meaning in software engineering and this meaning is often considered too restrictive for tools.

Section 2.0 keeps the general objective for determinism but rewords it in order to address the unexpected behavior of the tools: "For a tool whose output may vary within expectations, it should be shown that the variation does not adversely affect the intended use of the output and that the correctness of the output is established."

1.6.2.3 TOR Clarification

This document clarifies when a TOR is needed and what it should contain. Objective 2 of Table T-0 (which references 5.1.1) provides an objective for the TOR. The details of the content of the TOR are provided in section 10.3.1.

1.6.2.4 Tool Integration in the Operational Environment

A new objective is defined for tools (objective 3 of Table T-0 (which references 5.3.1)). It ensures that the qualification occurs within a specific tool operational environment.

1.6.2.5 Tool Verification

Objective 5 of Table T-0 (which references 6.2.1.b) states: "Tool operation complies with the Tool Operational Requirements," and corresponds to section 12.2.2 of DO-178B. The intent of this objective is to ensure that the tool installed in the tool operational environment is compliant with the TOR.

1.6.2.6 Tool Validation

In addition to the verification process assessing that the tool is compliant to the TOR, two new objectives, corresponding to a validation activity, are added. These objectives are intended to assess that the tool is compliant with the needs of the airborne software life cycle:

- Objective 6 of Table T-0 (which references 6.2.1.aa): "Tool Operational Requirements are sufficient and correct." The aim is to create a tie between the TOR and the objective satisfied by the use of the tool. Compliance to this objective demonstrates that the requirements applicable to the tool cover the airborne (or CNS/ATM) software objective (fully or partially) satisfied by the use of the tool.

- Objective 7 of Table T-0 (which references 6.2.1.bb): "Software life cycle process needs are met by the tool." This objective is complementary to objective 6 of Table

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

T-0. It assesses that the tool, installed in the tool operational environment, satisfies all of the needs of the software process.

1.6.2.7 Tool Configuration Management

For TQL-5, the required qualification data is identified as CC2. Therefore, as a minimum, the following objectives identified in Table T-8 are applicable:

- Configuration items are identified
- Archive, retrieval, and release are established.

1.6.2.8 Tool Quality Assurance

The general guidance referenced by objective 2 of Table T-9 is applicable for all TQLs. Additionally, objective 5 of Table T-9 enforces the need for a review, but considers that this review may be supplemented or performed in the context of the airborne (or CNS/ATM) software conformity review. Other objectives of Table T-9 are not applicable to TQL-4 and TQL-5

1.6.2.9 Certification/Approval Liaison Process for Tool Qualification

All Table T-10 objectives apply to TQL-5 tools. The data that would normally be presented in a TQP or TAS may be provided in the PSAC (or PSAA for DO-278A users) and SAS of the airborne software (CNS/ATM software for DO-278A users).

To ensure that the tool performs its intended function within the software life cycle, even with known problems and functional limitations, an objective is added in Table T-10 and is applicable to TQL-5: "Impact of known problems on the Tool Operational Requirements is identified and analyzed."

1.6.3 Conclusion

This document provides more accurate and complete guidance for tools at TQL-5 than DO-178B (and hence DO-278) did for verification tools. The intent is not to ask for more activities or more data (for example, the qualification does not require any data from the tool development process). However, it clarifies the content of the TOR, the compliance of the tool to the airborne (or CNS/ATM) software process needs, and the objectives of other integral processes applicable for TQL-5.

Note: Since DO-278 references DO-178B with respect to tool qualification, all references to DO-178B are also applicable to DO-278 in this FAQ.

1.7 FAQ D.7: How Might One Use a Qualified Tool to Verify the Outputs of an Unqualified Tool?

Reference: This document: Entire document
DO-178C/DO-278A: Section 12.2

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Keywords: tools; tool qualification; qualified tool; unqualified tool

Answer:

1.7.1 Scope and Limitations

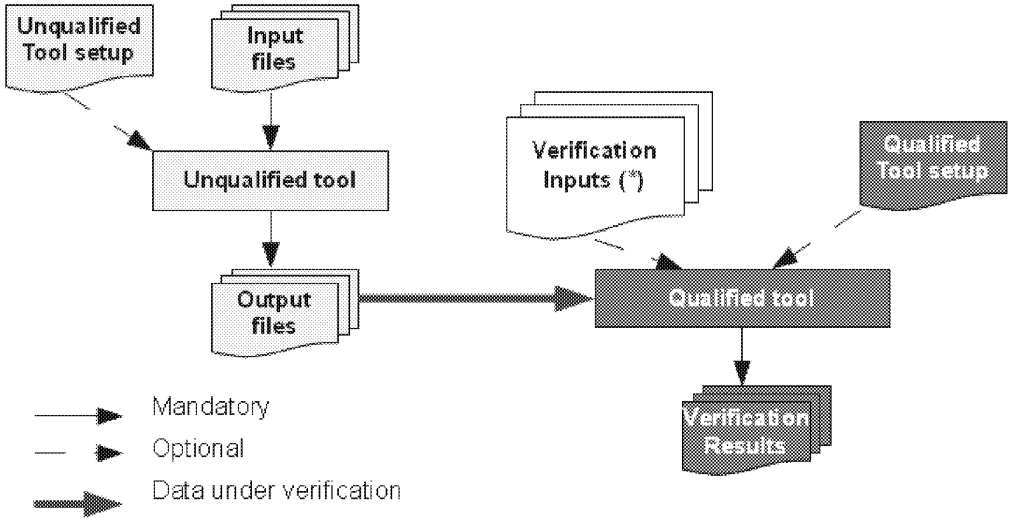
When the outputs of a tool are verified as described in DO-178C/DO-278A section 6, qualification of the tool is not required. This verification may be performed by a combination of reviews, analyses, and automated verification performed by one or more other tools to satisfy all objectives applicable to the unqualified tool outputs. The ability of tools to satisfy these objectives is key to selecting the overall verification approach.

For some tools, such as autocode generators, the satisfaction of all applicable verification objectives from DO-178C/DO-278A of the tool outputs with another tool may not be achievable. Therefore, a determination needs to be made about which objectives can be addressed by a qualified tool.

Per DO-178C/DO-278A and this document's guidance, the PSAC (or PSAA for DO-278A users) clearly explains and justifies the objectives, partially or fully, satisfied by the use of the tools and the need for qualification. The tool qualification criteria defined in DO-178C/DO-278A section 12.2 are applied to the verifying tool(s) in order to define its qualification level.

Figure D-1 shows an example of relationship between the tools.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.



(*): Data needed to verify the unqualified tool output files.
These data may include the unqualified tool input files

Figure D-1 Tool Relationship Example

This FAQ identifies some factors to be considered in order to address errors in both the unqualified tool and the qualified tool(s). Two cases are possible:

1. The unqualified tool (for example, autocode generator) injects an error in the airborne (or CNS/ATM) software, which the qualified tool fails to detect.

2. The outputs of the unqualified tool (for example, test case generator) are erroneous or incomplete and the qualified tool fails to detect it.

This FAQ is not intended to encourage or discourage the use of such a software life cycle, but is intended to identify aspects that need to be considered in order to avoid the two cases above. Specific details are documented in the certification/approval plans and coordinated with the certification/approval authority.

1.7.2 Discussion

This FAQ addresses the following considerations regarding the possible problems raised when using a tool(s) to verify the outputs of an unqualified tool:

- Coverage of verification objectives that apply to the unqualified tool's output.
- Operating conditions of the qualified tool.
- Common cause avoidance (that is, avoiding a single error affecting both the unqualified tool and the qualified tool).
- Protection between the tools (that is, avoiding interference of the unqualified tool on qualified tool's proper operation).

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

Each of these is considered in greater detail below. In addition, other considerations may be raised for a specific life cycle or input/output data configuration, and need to be addressed case-by-case in the scope of the project.

1.7.2.1 Coverage of Unqualified Tool Output Verification Objectives

The activities and processes automated by the unqualified tool dictate the applicable verification objectives to be satisfied by the qualified tool. It is necessary to consider all applicable verification objectives in order to define the qualification effort. The PSAC (PSAA for DO-278A users) identifies which objectives are fully or partially satisfied by the qualified tool. This data may be further developed in a separate TQP referenced in the PSAC (PSAA for DO-278A users). It should be noted that the PSAC (PSAA for DO-278A users) also identifies how all applicable objectives are fully satisfied.

When the qualified tool is used to verify only an intermediate output of the unqualified tool, additional activities need to be performed on the final output of the unqualified tool.

It is important that the Tool Operational Requirements for the qualified tool is consistent with the objectives claimed in the PSAC (PSAA for DO-278A users), as required by Table T-0 (objective 6).

1.7.2.2 Operating Conditions of the Qualified Tool

The tool verifying the outputs of the unqualified tool is qualified for use in "normal operating conditions" defined in the tool qualification data. These conditions may include some constraints, limitations, and context on the unqualified tool usage. This context includes but is not limited to configuration and initialization settings (tool setup).

It is necessary to verify the compliance to these conditions when the qualified tool is used.

1.7.2.3 Common Cause Avoidance

A separation between the qualified tool and the unqualified tool is needed in order to avoid the presence of a similar error in both tools. This separation may take one of several forms, including but not limited to those listed below:

- Independent tool development: The qualified tool is developed independently from the unqualified tool, using separate development personnel within a separate tool development life cycle. Demonstrating the independence of tool development may require more tool development life cycle data than is normally necessary for qualification. This approach to independence does not necessarily require that the qualified tool be created by a different organization than the unqualified tool.

- Dissimilar technical approach: Depending upon the unqualified tool functionality, the qualified tool may need to use a different technical approach than the unqualified tool. This is similar to error-checking steps commonly used in solving algebraic equations, where one mathematical process is used to arrive at an answer, and a second process is used to check that the answer is correct.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

Regardless of the approach for separation, common code components (such as libraries) need to be specifically addressed.

1.7.2.4 Protection between Tools

The glossary of this document defines protection as: "The use of a mechanism to ensure that a tool function cannot adversely impact another tool function." If a qualified tool is used to verify the outputs of an unqualified tool, then the degree to which the unqualified tool may interfere with or adversely impact the qualified tool's proper operation needs to be minimized.

1.8 FAQ D.8: How Might One Use a Qualified Autocode Generator?

Reference: This document: Entire document
DO-178C/DO-278A: Section 12.2 and Annex A

Keywords: autocode generator; tool qualification

Answer:

1.8.1 Terminology

For purposes of this FAQ, the following terms are applicable:

- **Autocode Generator (ACG)**: This term covers all of the software tools that use a set of computer files representing a part of software requirements, such as a model, a formal specification, or a set of data. The ACG uses these files to generate "Source Code". The translation rules and the description of the ACG's input and Source Code format are provided in its Tool Operational Requirements and/or in the Tool Requirements.

- **Qualified ACG**: ACG qualified under DO-178C/DO-278A section 12.2, criteria 1 (since it could insert an error into the airborne (or CNS/ATM) code).

- **Input files**: Input of the ACG, considered as the low-level requirements for the airborne (or CNS/ATM) software, in the sense that Source Code can be directly implemented without any further information.

- **Compiler/linker:** Tools used to translate Source Code into Executable Object Code, which is the format that can be integrated on the target.

1.8.2 Purpose and Limitations of This FAQ

The correctness and the completeness of the input files, including the absence of unintended elements, are addressed by the software verification process and are outside the scope of this FAQ. The input files may be formatted as a model. In this case, the Model-Based Development and Verification Supplement needs to be consulted.

An ACG, as any tool, is qualified in the scope of a specific context identified in the PSAC (PSAA for DO-278A users). This context includes the identification of objectives satisfied through the use of the qualified tool.

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

The purpose of this FAQ is to clarify under which conditions some certification/approval credit (satisfaction of objectives) may be claimed when using a qualified ACG. However, it is not possible to address all technical issues when using a qualified ACG. This FAQ provides additional information based on some typical scenarios, including constraints and limitations, when seeking credit using an ACG. These scenarios provide insight into the thought process and potential considerations to be addressed when using a qualified ACG. The constraints, limitations, and considerations identified in this FAQ are not exhaustive.

1.8.3 Discussion

This FAQ discusses how some objectives of the airborne (or CNS/ATM) software development processes may be satisfied through the use of a qualified ACG. The following DO-178C/DO-278A objectives areas are addressed: Source Code verification (see 1.8.3.1 below), Executable Object Code verification (see 1.8.3.2 below), and verification of output of software testing (see 1.8.3.3 below). Section 1.8.3.2.1 provides scenarios of applying the concepts presented in sections 1.8.3.2 and 1.8.3.3. Figure D-2 provides a point of reference for the FAQ.

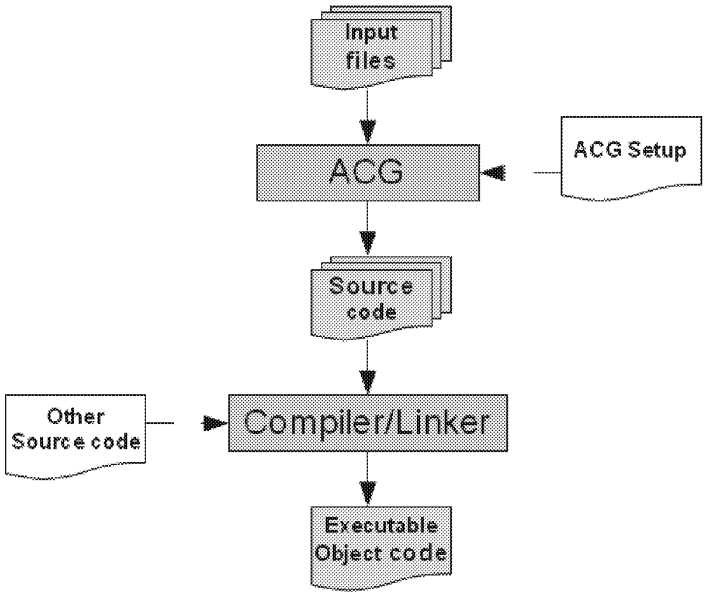


Figure D-2 Development Process with an ACG

1.8.3.1 Source Code Verification Objectives

Qualification of the ACG may allow one to reduce, automate, or eliminate some of the recurrent Source Code verification processes (objectives 1 to 6 of DO-178C/DO-278A Table A-5), with the non-recurrent activity of tool qualification. This section reviews the

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

characteristics of the ACG qualification data necessary to enable the satisfaction of verification objectives relevant to the Source Code.

As required in this document, all the translation rules from input files to the Source Code are defined in the Tool Requirements. The correctness of the translation rules is verified through Tool Requirements verification, and the correctness of their implementation is verified through the tool testing process and the tool operational verification and validation process. This includes verifying that Source Code includes only elements identified in the Tool Requirements. Provided that the Tool Requirements are accurate and tool verification activities are complete and relevant to the Source Code verification objectives, credit may be claimed for DO-178C/DO-278A Table A-5, objectives 1 to 6, with the following limitations:

- Part of objective 6 of Table A-5, worst-case execution time or stack usage analyses, may only be satisfied after the Source Code generation.
- It is also required to verify that the tool has been exercised on the complete set of input files to ensure that all the low-level requirements have been developed into Source Code (Table A-5, objective 5).

1.8.3.2 Executable Object Code Verification Objectives

The correctness of the Executable Object Code is classically verified through requirements-based tests (DO-178C/DO-278A Table A-6), and the related verification of verification data (DO-178C/DO-278A Table A-7).

Even when using a qualified ACG, a large part of the Executable Object Code verification remains outside the scope of the ACG qualification. This includes all of the following:

- Hardware/software integration testing.
- Software/software integration testing, particularly the integration between ACG generated code and other software parts.
- Requirements-based tests for compliance of the Executable Object Code to the software requirements above those provided as input to the ACG.
- Requirements-based tests for libraries that may be called by the ACG generated Source Code.
- Compiler analysis as described in section 4.4.2 of DO-178C/DO-278A.
- Additional verification on additional code generated by compiler, linker, or other means not directly traceable to Source Code.

Three potential scenarios demonstrate how the activities performed in the software life cycle and in the tool qualification process may satisfy DO-178C/DO-278A Table A-6 objectives 3 and 4 (that is, Executable Object Code complies with (normally and robustly) the low-level requirements).

1.8.3.2.1 Scenarios Using the ACG

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

This section describes how a qualified ACG may be used in the airborne (or CNS/ATM) software development process. The following scenarios highlight how DO-178C/DO-278A objectives may be satisfied by the qualified tool or through the airborne (or CNS/ATM) software process:

- Scenario 1 - Satisfaction of low-level requirements-based tests objectives through tests cases based on the low-level requirements.
- Scenario 2 - Satisfaction of low-level requirements-based tests objectives through tests cases based on the requirements from which the model (input files) is developed.
- Scenario 3 - Satisfaction of low-level requirements-based tests objectives through qualification of the ACG and verification of a set of representative input files.

Implementation of these scenarios into an actual project should be closely coordinated with the certification/approval authority.

Table D-4 provides an overview of the objectives that may be satisfied for the three scenarios given, with the following notations:

- "Tool" means that it may be satisfied through tool qualification process.
- "Software" means that it may be satisfied through airborne (or CNS/ATM) software processes.
- "Tool/Software" means that it may be partially covered by tool qualification process and partially by airborne (or CNS/ATM) software processes, as described in the following sections.

Table D-4 Overview of Some ACG Use Scenarios

Objectives	DO-178C/ DO-278A Table A-5 (Obj 1-4)	DO-178C/ DO-278A Table A-5 (Obj 5, 6)	DO-178C /DO-278A Table A-6 (Obj 3, 4)	DO-178C /DO-278A Table A-7 (Obj 1, 2, 4)	DO-330 Table T-0 (Obj 5, 7)	DO-178C /DO-278A Table A-7 (Obj 5-7)
Scenario 1	Tool	Tool/Software	Software	Software	Software (Note 2)	Software
Scenario 2	Tool	Tool/Software	Software	Software	Software (Note 2)	Tool/Software
Scenario 3	Tool	Tool/Software	Tool	Tool/Software	Tool	Tool/Software

Note 1: DO-178C/DO-278A objectives not identified in this table are typically not impacted by the use of an ACG.

Note 2: These scenarios show that data produced during the airborne (or CNS/ATM) software development process may be used to satisfy tool qualification objectives. This concept will be further explained below.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

Scenario 1: Satisfaction of low-level requirements-based test objectives through test cases based on the low-level requirements.

In this scenario the "credit" claimed by the qualified tool is limited to the Source Code verification alleviation (that is, objectives 1 to 6 of DO-178C/DO-278A Table A-5). Low-level requirements-based testing is performed as part of the airborne (or CNS/ATM) software life cycle process (not through the tool qualification process). The low-level requirements-based testing of the airborne (or CNS/ATM) software ensures that the "software life cycle process needs are met" and that "the tool operation is compliant to the Tool Operational Requirements."

In this scenario, one may propose to satisfy the two following objectives of the tool qualification process in the scope of the airborne (or CNS/ATM) software process rather than in the scope of tool qualification process:

- Tool Operation complies with the Tool Operational Requirements (objective 5 of Table T-0)
- Ensure software life cycle process needs are met (objective 7 of Table T-0)

Scenario 2: Satisfaction of low-level requirements-based test objectives through test cases based on the requirements from which the model (input files) is developed.

The input files of an ACG may be developed in the form of a "design model". The Model-Based Development and Verification Supplement applies to this design model and includes the following activities:

- Design model is developed from a set of requirements.
- Model verification is performed through a combination of reviews, analyses, and simulations.
- Tests are developed from the requirements from which the model is developed and executed. These test data are also verified to satisfy objectives 1 and 2 of DO-178C/DO-278A Table A-7.
- Additional tests (specifically for derived requirements contained in the model) are developed and executed.
- A model coverage analysis is performed through verification activity (tests or simulation) based on the requirements from which the model is developed.

If the following conditions on the completion of these activities are satisfied, in particular through the satisfaction of the coverage of the requirements contained in the model (objective 4 of DO-178C/DO-278A Table A-7), it may not be necessary to duplicate the tests for requirements contained in the model, as noted in section DO-178C/DO-278A section 6.4:

- The verification cases used for model coverage analysis are equivalent to test cases.
- The analyses of these test cases demonstrate the coverage of requirements from which the model is developed.

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

- The analyses of these test cases demonstrates the coverage of requirements contained in the design model (the inputs files) through the model coverage analysis, assuming this analysis demonstrates normal range and robustness criteria are also satisfied.

Therefore, it may be claimed that the compliance of Executable Object Code to the Design Model (the input files) objectives are satisfied.

The requirements-based tests ensure that the airborne (or CNS/ATM) software life cycle process needs are met and that the tool operation is compliant with the Tool Operational Requirements. Therefore, one may propose to satisfy the two following objectives of the tool qualification process through the airborne (or CNS/ATM) software process instead of the tool qualification process:

- Tool operation complies with the Tool Operational Requirements (objective 5 of Table T-0).

- Ensure software life cycle process needs are met (objective 7 of Table T-0).

Scenario 3: Satisfaction of low-level requirements-based test objectives through qualification of the ACG and verification of a set of representative input files.

1. **Executable object code complies with low-level requirements (objective 3 of DO-178C/DO-278A Table A-6)**

   The objective of compliance of Executable Object Code to the input files may be satisfied in the scope of the tool qualification process. It is the purpose of the tool operational verification and validation process, including:

   - The definition of a set of representative input files that include, but are not limited to the following:
     - All the allowed elements (for example, units from which a model is constructed) in accordance with the applicable standards.
     - Representative combinations of those elements (equivalence classes).
     - Limits of the applicable inputs standards.
     - The allowed complexity of inputs.
   - The execution of the ACG on the set of input files and the generation of Source Code.
   - The generation of the Executable Object Code by using the same generation environment used for the airborne (or CNS/ATM) software, including the same compiler/linker with the selected options.
   - The verification of the compliance of the Executable Object Code to the representative input files.

   This approach may be proposed as equivalent to low-level requirement-based tests, performed through equivalent classes of input files. The activity performed in the scope of the tool operational verification and validation process includes,

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

as a minimum, the development of test cases and procedures; execution of the tests procedures to obtain the test results; and the verification of all these test data to satisfy objectives 1, 2, and 4 of DO-178C/DO-278A Table A-7. To satisfy objective 4 of DO-178C/DO-278A Table A-7 a coverage analysis of the requirements contained in the representative set of input files needs to be performed.

The representative set of inputs files are assessed through the verification of the test data, including:

- The demonstration of coverage of the actual possible input files. This is performed through verification that the representative set of input files includes: (1) all the allowed elements, (2) an acceptable degree of combination of allowed elements representative of all combinations that may be used in airborne (or CNS/ATM) software, and (3) the size and complexity limits.

- An additional analysis needs to be performed on the generated Source Code to assess that it includes all possible Source Code statements that may be generated by the ACG.

In order to make this analysis easier, precautions and constraints are defined on input files and on the generated Source Code, as follows:

- The set of allowed elements (for example, units from which input files are constructed), the number of elements in each input file, and the allowed combinations may be reduced through the "input files" standard. For example, in the case where input files are given in the form of models this may impose modeling restrictions that would be part of the airborne (or CNS/ATM) Software Design Standards.

- The possible set of statements in the Source Code and also the complexity of their combinations need to be reduced as much as possible. Reduction is necessary to ease the demonstration that the possible errors that may be introduced by the compiler/linker will be detected during the tool operational verification and validation process. This reduction is achieved during the Tool Requirements process that defines the Source Code to be generated. For example, complex algorithms could be developed in separately verified libraries, and included in the generated Source Code through external calls. Another example is to reduce the number of nesting levels in the Source Code.

2. Executable Object Code is robust with low-level requirements (DO-178C/DO-278A Table A-6, objective 4)

One may propose to satisfy this objective through a combination of an analysis and tests. The analysis includes:

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

- Assessment of completeness of robustness requirements included in the software requirements above those input to the ACG.
- Existence and coverage of Design Standard rules applicable to the input files to insert some robustness mechanism inside the input files.
- Verification of the input files' compliance to the Design Standards, including the robustness mechanism rules.
- Identification of defensive programming techniques to be inserted in the generated Source Code by the ACG. These mechanisms need to be identified through the ACG Tool Requirements.

Additional tests may be developed to supplement this analysis, if necessary.

1.8.3.3 Coverage of Software Structure Objectives

According to DO-248C FAQ #43, the purpose of structural coverage analysis is to:

1. provide evidence that the code structure was verified to the degree required for the applicable software level;
2. provide a means to support demonstration of absence of unintended functions; and
3. establish the thoroughness of requirements-based testing.

The text below explains how a qualified ACG may satisfy each of the three purposes.

1.8.3.3.1 Verification of airborne (or CNS/ATM) code structure

Verification of airborne (or CNS/ATM) software code structure may be obtained through the use of the qualified ACG, to the extent demonstrated by the tool qualification process. Tool Requirements for the qualified ACG define the Source Code generated for all the allowed inputs. Therefore, the tool verification process confirms that:

- The Tool Requirements are correct and identify all possible code structures that can be generated from the inputs permitted by the airborne (or CNS/ATM) software Design Standards.
- All the generated Source Code structures are implemented as defined in the Tool Requirements.

1.8.3.3.2 Absence of unintended functions

In this section, "unintended functions" should be interpreted as functions strictly related to the process of translation from input files to Source Code. The main purpose of the ACG qualification process is to demonstrate that the Source Code exactly implements the input files.

Thus, the presence of unintended function attributable to the tool operation in the generated Source Code is precluded through the use of the qualified ACG to the extent demonstrated by the tool qualification process:

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

- The Tool Requirements are correct and are accurate enough to verify that no unintended functions are generated.

- The generated Source Code implements the input files as defined in the applicable Tool Requirements.

1.8.3.3.3 Thoroughness of requirements-based testing

Establishing the thoroughness of requirements-based testing requires a complementary activity, which will vary based on the above scenarios:

- In Scenario 1, structural coverage analysis may be performed through the verification of the low-level requirements-based tests.

- In Scenario 2, an analysis demonstrates that the model coverage analysis provides the same confidence as a coverage analysis performed at Source Code level. That may include an analysis that there is a direct traceability between model structure and Source Code structure.

- In Scenario 3, since the low-level requirements-based testing may be replaced by the tests performed in the framework of the tool operational verification and validation process, the thoroughness of requirements-based testing is obtained through the tool operational verification and validation process, including the evaluation that the set of inputs used are: (1) representative of the project requirements, and (2) sufficient to satisfy the requirements coverage objectives.

Note that in all scenarios, structural coverage analysis may be performed through the verification of the requirements-based tests performed against the software requirements above those input to the ACG.

1.9 FAQ D.9: Is Qualification of a Model Simulator Needed?

Reference: DO-178C/DO-278A: Section 12.2

Keywords: tool qualification; model simulator; simulation

Answer:

According to DO-178C/DO-278A section 12.2, the qualification of the model simulator, as for other tools, is needed "when processes of this document are eliminated, reduced, or automated by the use of a software tool without its output being verified as specified in section 6."

One purpose of simulation is to provide repeatable evidence of compliance of the model against the requirements from which the model was developed. Simulation consists of the development of simulation cases and procedures based on requirements from which the model was developed, and the subsequent execution of those simulation procedures in a specific environment.

Model simulation activities typically include:
1. Development of simulation cases based on requirements from which the model was developed.
2. Development of simulation procedures.

For Personal Use by Jordan Colson, Skyryse
© 2011 RTCA, Inc. and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.

3. Execution of the simulation procedures in the specific environment to obtain simulation results. The model simulator is the main tool of this specific environment.
4. Verification of simulation cases and procedures.
5. Verification of simulation results and explanation of any discrepancies. This includes the verification of the compliance of the model simulator outputs to the expected results defined in the simulation cases.

When the simulation is performed to satisfy some model verification objectives only, and the tool outputs (simulation results) are verified to be correct with respect to the inputs (expected results defined in the simulation cases and procedures), the applicant may propose to not qualify the model simulator. This proposal should be based on the simulation environment analysis and the thoroughness of the simulation data verification.

However, if the model simulator includes some functions that for example, automate the verification of the simulation results or the generation of the simulation procedures, these functions may fail to detect an error and should be qualified, if the outputs of these functions are not verified.

Additionally, section 11.1 of this document (about "multi-function tools") needs to be considered when a model simulator is used both for model verification and for other activities.

For Personal Use by Jordan Colson, Skyryse
and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details. © 2011 RTCA, Inc.

This Page Intentionally Left Blank

For Personal Use by Jordan Colson, Skyryse

and not for sale, transfer, or distribution to any other party. See Electronic License Agreement for more details.